FULL ENVELOPE CONTROL OF NONLINEAR PLANTS WITH PARAMETER UNCERTAINTY BY FUZZY CONTROLLER SCHEDULING

DISSERTATION
Thomas J. Kobylarz
Captain, USAF

AFIT/DSG/ENG/95S-05

19960327 041

DISCLAIMER NOTICE

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.

AFIT/DSG/ENG/95S-05

FULL ENVELOPE CONTROL OF NONLINEAR PLANTS WITH PARAMETER UNCERTAINTY BY FUZZY CONTROLLER SCHEDULING
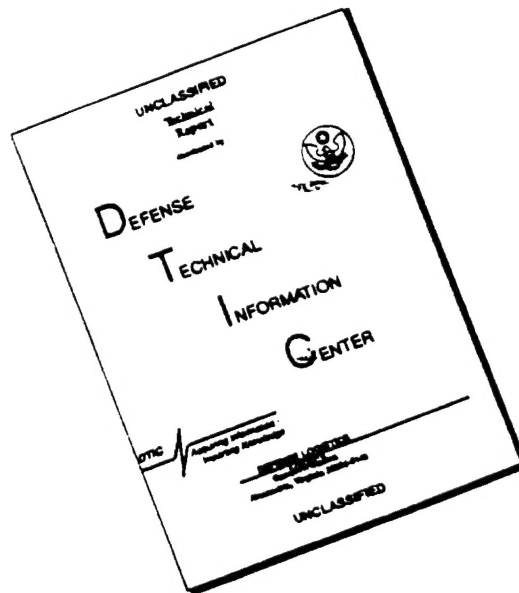
DISSERTATION

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

Thomas J. Kobylarz, BSEE, MSEE

Captain, USAF

September, 1995

AFIT/DSG/ENG/95S-05

FULL ENVELOPE CONTROL OF NONLINEAR PLANTS WITH PARAMETER UNCERTAINTY BY FUZZY CONTROLLER SCHEDULING

Thomas J. Kobylarz, BSEE, MSEE

Captain, USAF

Approved:

M. Pachter   24 Aug. 1995

Meir Pachter, Chairman

Constantine H. Houpis   24 Aug. 95

Constantine H. Houpis

            24 Aug '95

Steven K. Rogers

         24 AUG 95

Dennis W. Quinn

        24 Aug 95

Matthew Kabrisky, Dean's Representative

Accepted:

Robert A. Calico, Jr
Dean, Graduate School of Engineering

Acknowledgements

I would like to thank all my professors at AFIT for providing me with the foundations for completing this dissertation, especially my research committee. The endless reviews of Dr Meir Pachter and Dr Constaintine Houpis ensured meaningful and objective results. My thanks also goes out to all the other faculty and staff who in some way contributed to the completion of this document.

Never ending thanks goes out to my beautiful wife Korina for forcing me to sit down and complete this work, as well as taking care of the family in my absence. Of course, I can't leave out the kids; thanks Zac, Jake and Karlie for adding a little spice to life. Now we can finally spend more time together. To all the guys on the Hockey team, thanks for letting me take out my frustration.

Finally to my officemates, past and present, thanks for the help and ears.

Thomas J. Kobylarz

Table of Contents

Notation

Scalars, Vectors, Matrices

    Scalars. Denoted by upper or lower case letters in italic type.

    Vectors. Denoted by lower case letters in italic type, vector or scalar is taken from context. The n-dimensional vector x is made up of components x_i for $i = 1, \ldots, n$.

    Matrices. Denoted by upper case letters in boldface type, as the matrix $\mathbf{A}$, made up of elements A_{ij} (ith row, jth column).

Superscripts

$$
\begin{array}{rcl}
(\cdot)^T & : & \text{transpose (matrix)} \\
(\cdot)^{-1} & : & \text{inverse (matrix or transform)} \\
(\cdot)^* & : & \text{optimal solution, complex conjugate, or complement (set)} \\
(\dot{\cdot}) & : & \text{differeniation with respect to time}
\end{array}
$$

Operators

    Functions and mappings are set in an italic font.

Matrix and Vector Relationships

$$
\begin{array}{rcl}
\mathbf{A} > \mathbf{B} & : & \mathbf{A} \text{ - } \mathbf{B} \text{ is positive definite} \\
\mathbf{A} \geq \mathbf{B} & : & \mathbf{A} \text{ - } \mathbf{B} \text{ is positive semidefinite} \\
x \geq a & : & \text{component wise}, x_1 \geq a_1, x_2 \geq a_2, \ldots, x_n \geq a_n
\end{array}
$$

Sets

- A blackboard font denotes Algebraic Fields and Linear Spaces.

- A calligraphic font denotes Fuzzy Sets and Fuzzy Rules.

- A sans serif font denotes classical sets.

all of which will be upper case, some examples are:

$\mathbb{R}$: All real numbers

$\mathbb{R}^-$: $\{r \in \mathbb{R} \mid r \leq 0\}$

$\mathbb{R}^+$: $\{r \in \mathbb{R} \mid r \geq 0\}$

$\mathbb{N}$: All natural numbers

$\mathbb{C}$: All complex numbers

$\mathcal{M}$: A Fuzzy Set

$\mathcal{R}_i$: The ith Fuzzy Rule

A : $\mathsf{A} = [a, b] \subset \mathbb{R}$ where $a \leq b$; $a, b \in \mathbb{R}$

List of Figures

List of Tables

List of Symbols

xvii

List of Abbreviations

AFIT/DSG/ENG/95S-05

Abstract

A full envelope controller synthesis technique is developed for multiple-input single-output (MISO) nonlinear systems with structured parameter uncertainty. The technique maximizes the controller's valid region of operation, while guaranteeing pre-specified transient performance. The resulting controller does not require on-line adaptation, estimation, prediction or model identification. Fuzzy Logic (FL) is used to smoothly schedule independently designed point controllers over the operational envelope and parameter space of the system's model. These point controllers are synthesized using techniques chosen by the designer, thus allowing an unprecedented amount of design freedom. By using established control theory for the point controllers, the resulting nonlinear dynamic controller is able to handle the dynamics of complex systems which can not otherwise be addressed by Fuzzy Logic Control. An analytical solution for parameters describing the membership functions allows the optimization to yield the location of point designs: both quantifying the controller's coverage, and eliminating the need of extensive hand tuning of these parameters. The net result is a decrease in the number of point designs required. Geometric primitives used in the solution all have multi-dimensional interpretations (convex hull, ellipsoid, Voronoi/Delaunay diagrams) which allow for scheduling on n-dimensions, including uncertainty due to nonlinearities and parameter variation. Since many multiple-input multiple-output (MIMO) controller design techniques are accomplished by solving several MISO problems, this work bridges the gap to full envelope control of MIMO nonlinear systems with parameter variation.

FULL ENVELOPE CONTROL OF NONLINEAR PLANTS WITH PARAMETER UNCERTAINTY BY FUZZY CONTROLLER SCHEDULING

I. Introduction

1.1 Motivation

The vast majority of control design techniques are based upon a mathematical model of the system, or "plant", that is to be controlled. These models allow the use of analytical tools to guarantee that performance specifications will be met; but these guarantees only hold as long as the underlying models are valid. Thus, many systems require complex control strategies to perform their designed tasks, especially those control systems that are required to operate in an unstructured environment. Furthermore, dealing with the entire dynamic range of operation can bring a control design technique to its knees. This is where the true controller design problem lies. Varying parameters and uncertainty from sensor noise, disturbances and perhaps even failures, ensure that the model is never perfect. An example of such a problem arises in flight control, where one is dealing with the nonlinear dynamics of an aircraft, whose parameters, in addition, vary continuously over its entire flight envelope. Thus the problem is then two-fold. First, the nonlinearity/complexity of the model; and secondly, the variation, or uncertainty in the model's parameters. In conclusion: The former problem is encountered when large amplitude slewing maneuvers are attempted. The latter poses problems when operating in an unstructured environment is required.

Perhaps the most useful way of dealing with nonlinearity of the model is to linearize it about some point, p, in its operating range; that is about a point in the

parameter space, P, of the model. The parameter space P encompasses both the set of generic parameters that govern the dynamics of the plant, and variables which define the operating condition in the operational envelope. If the model is "smooth", a rather unrestrictive assumption for many physical systems, the linearized equation will accurately represent the true system in some "sufficiently small" region, or ball $B(p, \epsilon)$, about the equilibrium point p in the parameter space. The scalar ϵ represents how far the actual operating point can deviate from p and still be "adequately" described by the model and is determined by the strength of the pertinent nonlinearity. One now has available all the tools for linear analysis, and the solution within this neighborhood can be obtained by a myriad of linear control synthesis techniques, i.e. LQR, QFT, etc. However, one must still deal with varying parameters over the entire operating range. Varying the model's parameters may "remove" the system from within this region of model validity. The controller achieved above may yield nevertheless acceptable performance beyond the region for which it was designed, but this must be construed as luck in a specific problem solution. In an attempt to ensure adequate performance over the entire parameter space, the designer must adequately cover the entire space with a valid region, or regions, upon which to base the design. Robust controllers are those which attempt to increase the volume of such a region in the parameter space. One robust control design technique that actually *quantifies* its valid design region is Quantitative Feedback Theory [8, 16].

Frequently no single controller will do. A common practice is to perform several point-wise control designs, each design performed for a fixed $p \in$ P, that will adequately cover the entire operational range. These point designs need not be designed for only one point, but may be robust controllers covering a specified region of P. For instance in QFT where the region of acceptable performance in P is specified. Such robust controllers are considered to be designed *around a point* in P and therefore, will also be referred to as point designs. Classically, this requires overlap of the balls indicating the valid regions of the individual models. The rationale is then that

for any fixed point in the parameter space, one chooses "the best" controller and uses it. In this type of approach, one must devise a means of smoothly switching between controllers without inducing an objectionable response during the transition. This can be interpreted in a broad sense as "robust scheduling"; in particular "gain scheduling" is when the (not necessarily robust) controllers are of a common parametric form and these parameters are scheduled.

The heart of using multiple point designs is three fold. One must devise means to: 1) select the locations of the point models at which point designs are generated, 2) choose the best controller among those available and, 3) smoothly switch between controllers. The resulting controller can work quite well in many cases, as has been proven in flight control for years. However, the means by which the actual scheduling between the point designs is accomplished is mainly art and very ad hoc [34]. All three of these steps are "problem areas" which are over come in a systematic and quantified manner in this research.

1.2   Research Direction

This research effort focuses on the judicious scheduling of individual point designs (which may be robust with specified operating regions) over all of **P**. The parameter space consists of the actual physical parameters and/or the system's state about which the linearization is performed. The proposed approach is based upon using Fuzzy Logic (FL) to blend the individual "point" designs such that for any trajectory in the parameter space, the system performs (controls) adequately. The ability to systematically design such a *dynamic* scheduler is a major contribution to the field of controller design.

Fuzzy Logic is a partial membership set theory developed by Lotfi Zadeh in the mid 1960s and is basically a means of representing uncertainty in a system process without directly applying statistical methods [21]. Fuzzy Logic is now being used in

many academic fields and in commercial endeavors, and may be directly employed to: [21, 48].

1. Design a controller for crisp nonlinear or uncertain plants.

2. Perform System Identification (ID) of a plant.

3. Model an uncertain plant mathematically.

The goal of this research is to derive and explore a technique to design full envelope controllers, for nonlinear plants with structured parameter uncertainty, using point-wise designs that adequately span the parameter space of the plant to be controlled. The ability to base the controller on point-wise designs allows the designer to use all the available tools of classical, modern and robust control theory to aid in the solution. The term "envelope" is taken from the flight control field and it represents that subset of P defined by the (structural and aerodynamic) physical limitations of the airframe (plant). Where robust control's aim is to increase the valid region in P for a fixed compensator design, adaptive or scheduled control modifies the controller based upon an estimate of the current operating point in P. Thus, the efforts of this research is to develop a type of adaptive controller, based on scheduling on "fast" states. Successful development of such a technique is a significant contribution to the field of applied adaptive control.

The application of the Fuzzy Logic methodology yields a nonlinear mathematical problem. Thus, the mathematics required to analyze the problem and arrive at a solution reside in the field of nonlinear analysis and quickly become intractable. Classical analytical methods of guaranteeing the stability and performance of the control system are no longer applicable. Hence, this research is somewhat exploratory and it will rely to a point on heuristics and extensive simulations. This stage of affairs is a major drawback of the investigated FLC approach.

This research focuses on the following.

- The development of a multivariate Fuzzy Logic control paradigm. This will afford state feedback control in a fuzzy setting.

- Effects of switching between independent point designs as the plant traverses P.

- How to correctly/optimally blend independent point designs as the plant traverses P while maintaining acceptable performance.

- Does the ability to blend the individual designs impose any restrictions on the point designs themselves? That is, can the point designs be accomplished independent of each other (highly desirable), and may any conventional control design technique be used to achieve each separate point design? To accomplish this, the interaction of the control design method and the fuzzy blending of the point designs is investigated.

- Does this blending ability of the proposed technique provide any characteristics which relieves constraints on the underlying point designs? If so, this may allow for simpler methods of point design controller synthesis (i.e. plant inversion based techniques) that would be unacceptable without the addition of the fuzzy scheduling. Also, does the blending extend the valid region for which a controller may be used. That is, will the blending allow for a decrease in the number of point designs that would otherwise be required to cover P.

- Examination of what is a sufficient cover of P.

- Conduct extensive testing via simulation to evaluate the performance of the final Fuzzy Logic Controller (FLC).

1.3   Control System Description

This research entails blending individual point designs via Fuzzy Logic to achieve acceptable responses over the entire envelope of operation. To visualize
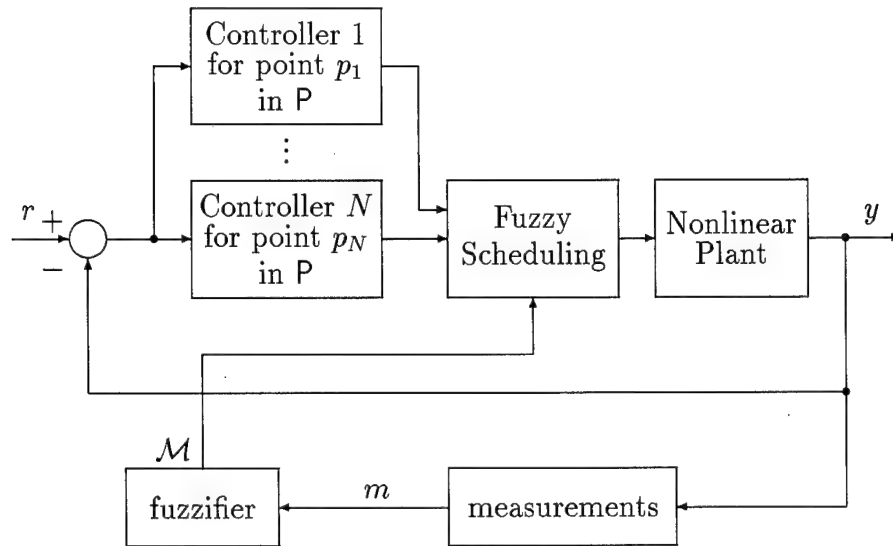
Figure 1.1   Functional Diagram of Adaptive Controller Using Fuzzy Scheduling

the concept, Figure 1.1 depicts a functional block diagram of the proposed adaptive control system.

A "sufficient" number of individual controllers are designed a priori, say N, such that $\bigcup_{i=1}^{N} B(p_i, \epsilon_i) \supseteq \mathsf{P}$. The parameter ϵ_i denotes the range of applicability of each point design p_i. Classically this requires overlap of the point controllers' valid regions such that for any $p \in \mathsf{P}$ there exists at least one point design yielding adequate response. The vector m, $m \in \mathsf{P}$, consists of available measurements of parameters and states on which the model relies. This measurement is then fuzzified to account for the uncertainty of the unknown true parameter values. The fuzzified m defines $\mathcal{M}$, a fuzzy set defined on the "universe of discourse" P. This represents the uncertain point of operation in the parameter space. Then the appropriate controllers are blended, based on $\mathcal{M}$, to drive the plant.

1.4 Research Scope and Assumptions

A technique is generated for the synthesis of the blending function of Figure 1.1 for acceptable control system performance over the entire operational envelope. Embedded in this goal is a determination of what is a sufficient covering of the envelope by point designs. This removes the requirement for conventional gain scheduling in control problems in which a single fixed controller can not perform adequately. In the development of this technique, the ability to synthesize all individual point designs independently and by whatever means the control engineer prefers is maintained. This allows the greatest applicability, including techniques which require linear time-invariant (LTI) plant models such as output feedback, LQR or eigenvector placement.

Due to the nonlinear aspects of the plant under control and the introduction of fuzzy reasoning, the issues of stability, performance and steady-state errors are addressed through experimentation via simulation and compared to prespecified system response requirements.

Although the techniques used to design the point controllers may very well require the plant model to be smooth in some region about its trim (or equilibrium) point $p \in \mathsf{P}$, the blending technique does not have this requirement. For the fuzzy scheduling, only continuity of the plant model is assumed.

The proposed design approach is applicable to both uncertainty due to plant nonlinearity, and uncertainty due to parameter variation or mis-modeling. As such, the proposed research constitutes an effort in *both nonlinearity and parameter uncertainty*. When insufficient point-wise designs exist to cover P, the uncertainty due to nonlinearity is greatly increased and adequate performance can not be "designed in" by the point-wise controllers alone. Therefore, the issue of sufficient cover must be addressed. While heuristic arguments are employed, the proposed approach is strongly anchored in the conventional control paradigm.

1.5 Current Literature

The majority of the FLC research and development in the existing literature, especially the dynamic/adaptive designs, represents work by people with neural network backgrounds. The adaptive properties of such work usually comes from using adaptive networks that play a part in either the antecedent or consequent action of the fuzzy rule set. The adaptive designs not based upon neural networks rely on an optimization criterion to change the antecedent (membership function) or consequence (control output) of the fuzzy rules.

The current adaptive work can be put into categories based upon a few discernible common roots. The techniques differ mainly based upon one's belief in the availability of accurate analytical models describing the system to be controlled. The confidence in such models ranges from none, yielding techniques that rely on empirical input/output data using adaptive neural networks; to very strong, where state cell [42] approaches and dynamic programming [30] ideas are used to arrive at nonlinear controllers. In between these two extreme levels of confidence in the model are: 1) techniques based on fuzzy identification of the system, 2) those which believe the system is better modeled as a system with varying or unknown membership functions and, 3) those that optimize on the consequence of the fuzzy rule. Clarification of the above groupings is given below.

The main difference between ordinary adaptive neural networks and those of practical use in fuzzy logic controllers is the ability to incorporate linguistic rules given by a human expert [32, 44]. These controllers can also monitor the system's response to their past inputs to provide learning reinforcement as in Berenji's GARIC based controller [4, 5]. In the spirit of dynamic programming [30], fuzzy logic controllers have also been implemented in neural networks using temporal back propagation to modify the rules [18].

Another class of controllers uses fuzzy identification to obtain a model of the plant to be controlled. This model is then used within the controller in different

ways. One method is to use the model to predict the system's response to an original fuzzy rule. This response is compared with the desired response and the consequence of the rule is modified such that the new rule includes a correction to remove the predicted error [2, 23]. Another technique is to design a desired open loop controller and augment it with the "inverse model" dynamics obtained from fuzzy identification. The inverse model then "cancels" the actual plant and the response is dictated by the open loop controller [2, 22, 23]. Lai and Lin [22] combine this fuzzy identification method with modifying the rules' consequences via an optimization routine. They begin with a fixed set of membership functions to perform a fuzzy ID of the plant. The consequence of a control rule is taken as a linear combination of the fuzzy variables. The mean-squared error of the output is used as the minimizing performance index to solve for the coefficients of a rule's consequence. If the resulting "optimal" fit using a linear combination of the fuzzy variables is not "good enough", they use a complex search algorithm to change the membership functions defining the fuzzy variables. With these new membership functions they perform another identification. This process is continued until the desired fit is obtained [22]. Most techniques avoid this second degree of freedom in obtaining the final fuzzy inference engine.

Stepping up the level of confidence placed on analytical models, or perhaps just an expert's linguistic rules, are those techniques which assume a fixed rule set and a well defined performance evaluation cost function. The "degree of freedom" here is the membership function. The problems are posed using a parametric representation of the underlying membership functions. Examples include: for Gaussian functions use the of mean and variance [45], and for triangular membership functions use of the 2 points of support, and each triangle's center point [19, 22]. Various optimization algorithms can then be used to optimally select the parameters which yield membership functions giving the best performance. Some possible search techniques

for optimization include recursive least mean squares [45], complex searches [22], or Genetic Algorithms [19].

Very similar to the above techniques are those that begin with fixed membership functions and rules, then optimize by means of the control applied. That is, the form of the rule is fixed, but the exact consequence is not yet known. In these cases the antecedents are determined by all possible intersections of the membership functions. The rule is then a combination of basis functions. This is very similar to some fuzzy ID [22] techniques; the difference in using it for control is that these basis functions are now the available control inputs. The most common consequence is a linear combination of available inputs [40, 42]. An optimization is then performed, yielding the coefficients for each consequence, providing the entire rule set.

In the final category, the analytical model is used to design the controller. The problem is set in the fuzzy paradigm to either help in dealing with nonlinearities, model variation or noise. This is most often performed using a state cell approach [42, 40] which yields different optimal solutions depending on the location in the state space. Vachtsevanos shows how naturally fuzzy logic complements the cell state approach in problems with constrained inputs [42].

Another use has been the fuzzification of the LQR paradigm [38]. Fuzzy dynamics yield an optimal control law, $u = -\mathcal{K}x$ with fuzzy $\mathcal{K}$ which satisfies the Ricatti "inequality" equation. The control law dictates a membership function for all $\mathcal{K}$ which satisfy the inequality. The optimal $\mathcal{K}$ is the one which has maximum membership value in these membership functions [38].

There are also ad hoc nonlinear techniques. An interesting nonlinear technique guaranteeing stability and asymptotic tracking is based upon Lyapunov synthesis [43] and a nonlinear supervisor. Such techniques provide nice guarantees in steady-state yet provide little control over, or insight into, transient responses.

Fuzzy Logic has been used to perform classical gain scheduling of a proportional-integral-derivative (PID) controller for a linear single-input single-output (SISO) system [15, 50]. However, this scheme requires a common controller where only the gains are varied, and the problem is only posed in the SISO case. A fuzzy inference engine is used to obtain the three "adaptive" gains to out perform an optimal linear PID controller. Possible performance improvement over the linear PID is not surprising, since the FLC has the advantage of nonlinear control action. The proposition that this is actually equivalent to a fixed nonlinear PID controller is addressed in the sequel.

Fuzzy gain scheduling has also been used for LTI SISO systems using other point-wise techniques. Both pole placement by state feedback [49] and H_∞ [47] have been used to generate the underlying point-wise controllers. For the state feedback case, fuzzy logic with triangular membership functions interpolates the feedback gains between two point designs. The resulting controllers from the H_∞ point designs were replaced by "similar" controllers, all of the same form. Fuzzy Logic was then used to interpolate the poles and zeros of the two nearest point designs. Both designs rely on trial and error for selection of the placement of the point designs and linearity of the truth model, ignoring nonlinearities in the design. They also rely on the variation of only a single scalar parameter. Although not addressed in the paper, the H_∞ design is susceptible to adverse transients since a single controller is being used [25]. This is due to various dynamic controllers being switched in and out without proper handling of the current state of plant stored energy. That is, the initial conditions required of the controller states is not addressed [34].

There have been attempts at using Fuzzy Logic to blend two separate dynamic controllers [35, 36]. However, both controllers are for the same plant at one operating point and hence do not directly address uncertainty due to nonlinearities or parameter variation. The two controllers are designed using conventional techniques where neither design yields a satisfactory response by itself. One design yields a fast

but very lightly damped response, while the other has good transients but is too slow. Using Fuzzy Logic along with trial and error tuning, an acceptable response may be obtained, but again, only for the single operating point of the plant.

Fuzzy Logic is often hyped as being capable of providing control solutions for "difficult" to control plants, where other methods fail. A Fuzzy Logic Controller (FLC), it is claimed, does not require a mathematical model of the plant but instead is able to capture the expertise of an experienced operator. Thus, the control engineer encodes the operator's rules governing his actions into a fuzzy inference engine in the FLC to control the plant. Indeed it's conceivable that in "simple systems", where these rules are easily articulated, the method works well and converges to a suitable controller after some tuning of the inference engine during simulations [17].

Note the use of "simple system" versus "simple plant"; this is used to bring attention to the control objective. Take for example the classical inverted pendulum problem; deriving the equations of motion without small angle approximations yields a set of nonlinear differential equations which are not easily dealt with directly. Most control techniques would linearize the equations about the unstable equilibrium set point and proceed, while a fuzzy controller has no such requirements [17]. The inverted pendulum is not a simple plant. However, the pendulum can easily be kept at the inverted position by observing the state trajectory in the phase plane of the system, where the states are naturally the physical variables angle and angle rate. Fuzzy control rules can easily be written down from the physical phase plane analysis and fine tuned through simulation [17, 34].

A moments reflection upon the method of generating the fuzzy rules shows that the resultant controller is equivalent to a classical proportional-derivative (PD) type with nonlinear gain elements. A review of the history leading to the popular PID controller shows this means (of rule generation) was the prevailing method of controller design in the 1920s [3]. By observing skilled operators, it was shown the appropriate control to mimic the operator is the sum of three terms related to

the error, derivative of error (transients), and the integral of error (steady-state) [3]. This is confirmed by the wide applicability/acceptance of the PID controller. Conversely, it can be said that a conventional PID controller is a FLC [3]. In a FLC, the rules are used to generate a smooth mapping from error state to controller output. For example, a large error should receive a large corrective action. The desired mapping can therefore be accomplished by a nonlinear continuous map. As shown in Section 2.4.1 the fuzzy map may be viewed as an approximation to this nonlinear map, or vice-versa. Actually, in the field of fuzzy identification it has been shown that such fuzzy maps are dense in the space of real valued continuous functions on the universe of discourse [44]. Thus, the mystical performance gain over linear controllers attributed to FLCs is due to this nonlinear action. What is noteworthy here is that the nonlinear elements of the controller arise naturally within the FLC framework.

Hence, this type of system is referred to as "simple" since, albeit nonlinear, the dynamics are of low order and monotonic, so the control rules are simple to heuristically "figure out". Also, the plant can be adequately controlled using only the output error and the error rate. That is, in a "simple system" the plant has a sense of directionality and an increase in its system input directly translates into a corresponding increase in its output and vice versa. Now "order" is only an attribute of scalars. This is why there have been problems in designing a FLC for anything other than simple low order systems. A truly valid design technique must be capable of handling a system that is non-minimum phase or one that contains an inherent time delay, none of which are properly addressed. Most of the current FLC literature rely on such "simple systems" as examples, which unfortunately for FLC, are being advertised as complex.

In the case of more complex systems, the fuzzy rules are hard to determine. The very nature of the fuzzy rule, **IF** $\mathcal{A}$, **THEN** $\mathcal{B}$, implies that these rules are known and the FLC is merely implementing the known control law in a smooth

way. The current FLC literature is seriously lacking in the areas of multiple-input multiple-output (MIMO) and non-minimum phase or dynamic control problems.

In summary, the current literature is lacking in the very important areas addressed by this research. These include the ability to deal with high order systems, and systems possessing complex dynamics. This must be accomplished while dealing with the inevitability of a nonlinear system with varying system dynamics over its entire range of operation as well as structured parametric uncertainty. The techniques which have tried to address some of these issues have attempted controlling with fixed optimized gains or adaptive gains to meet the challenge which will not adequately control a large class of systems [8]. This research seeks a systematic means of adaptively including *dynamic compensation* to obtain an *acceptable control solution* for *nonlinear dynamic plants* in a *multivariable* context. A true and meaningful contribution to the field. Fuzzy Logic will be used to an advantage by means of its proven smoothing characteristics (see Chapter II) and heuristic appeal, not on claimed mystical problem solving capabilities. This should all be accomplished using optimization when meaningful and in a manner which yields insight to the control engineer throughout the design process, rather than arbitrary trial and error to obtain a solution. This will be accomplished via a fuzzy supervisory layer of control, allowing the designer freedom in choice of point-wise design tools at his disposal to aid in the overall design of a full envelope controller. Furthermore, a multivariate FLC theory is developed providing the tools for MIMO control.

1.6 Organization

This research consists of five chapters and supporting appendices. Chapter II develops the multivariate Fuzzy Logic paradigm. The classical scalar logic is generalized to a vector representation providing the basic tools to be used in the research. This provides the ability to address the multivariable control problem and advance a theory of fuzzy state feedback control. Chapter III develops the gen-

eral n-dimensional solution of fuzzy controller scheduling by properly posing the optimization to be performed along with a multi-dimensional example. Chapter IV contains detailed analysis of additional examples of a strongly nonlinear plant and a USAF C-135 lateral controller. Theses examples illustrate the goodness of the developed Fuzzy Logic controller scheduling paradigm, which is the object of this research. Finally, Chapter V offers conclusions and suggestions for further research.

In this research plant models are generated from analytic first principles as opposed to system identification, therefore their description is a set of differiental equations. For completion, in Appendix A, the utility of Fuzzy Logic for system identification is investigated along with examples of how Fuzzy Logic may be used to blend models together. Next, in Appendix B, preliminary experiments are performed to provide insight into many of the design decisions as well as point out areas that must be addressed by the final technique. Appendices D and C contain supporting material for Chapters IV and III respectively.

II. Multivariate Fuzzy Logic

In this Chapter the development of multivariate Fuzzy Logic is undertaken which draws upon previous work [34]. Fuzzy sets and fuzzy logic constitute the basis for fuzzy logic control.

2.1 Fuzzy Sets and Membership Functions

The description/specification of the underlying "fuzzy sets" is a crucial step in setting up any Fuzzy Logic problem and the subsequent synthesis of the proposed Fuzzy Logic based controller scheduling. Thus, the fuzzy sets construct allows for the measurements recorded by the sensors to be transformed into linguistic labels which feature in the preconditions of the "Expert System"-like rules. Using the conventions of Fuzzy Logic, the following terminology is maintained.

Let X be a set of objects. The "classical" set $A \subseteq X$ is defined as a collection of elements $x \in X$ such that each x either belongs to or does not belong to A. By defining a characteristic or membership function (MF) on each element of x, the classical set can be represented by a set of ordered pairs $(x, 0)$ and $(x, 1)$ representing non-membership and membership to the set A respectively. Unlike classical sets, *fuzzy sets* allow partial membership and indicate the *degree* of which an element belongs to the set. That is, their membership functions are not binary (or crisp) but multi-valued. Now the fuzzy set $\mathcal{A}$ can be represented by the set of ordered pairs

$$\mathcal{A} = \{(x, \mu_{\mathcal{A}}(x)) \mid x \in X\}$$

where $\mu_{\mathcal{A}}$ is the *membership function* defined on X and X is referred to as the *universe of discourse*. For each membership function defined on the universe of discourse, associate a *fuzzy variable* (FV) which takes on values (linguistic labels) for each. For example, suppose there are two fuzzy variables "x is small" and "x is BIG" with

respective membership functions μ_s and μ_b. Then each x in the universe of discourse is "small" to degree $\mu_s(x)$ and "BIG" to degree $\mu_b(x)$.

A (multivariable) fuzzy set $\mathcal{A}$ is a pair specified by:

- Its "support", which is a classical set $\mathsf{A} \subseteq \mathbb{R}^n$ that encompasses the so called "universe of discourse", and

- A non-negative membership (or weight) function μ (whose support is the set A), where,

$$\mu : \mathsf{A} \quad \rightarrow \quad \mathbb{R}^+$$
$$\mu(x) = 0 \quad \text{for} \quad x \notin \mathsf{A}$$

If in addition

$$\max_{x \in \mathsf{A}} \{\mu(x)\} = 1$$

then the weighing function μ is referred to as "normalized".

In this research, a class of *multivariable* membership (or weighing) functions, which are based on the Multivariate Gaussian distribution, is used. This is motivated by the analytical properties of the Gaussian function.

Thus, consider the multivariable fuzzy set $\mathcal{A}_i$. The normalized membership function $\mu_{\mathcal{A}_i}(x)$ is

$$\mu_{\mathcal{A}_i}(x) = e^{-\frac{1}{2}(x-\bar{x}_i)'\mathbf{R}_i^{-1}(x-\bar{x}_i)} , \quad \forall x \in \mathbb{R}^n \tag{2.1}$$

The weighing function $\mu_{\mathcal{A}_i}(x)$ completely characterizes the fuzzy set $\mathcal{A}_i$. In other words, the underlying "support" set $\mathsf{A}_i \subset \mathbb{R}^n$ of the fuzzy set $\mathcal{A}_i$ is explicitly parameterized by its "center point" $\bar{x}_i \in \mathbb{R}^n$ and by its "size", which is determined by the square roots of the eigenvalues of the real symmetric and positive definite $n \times n$ matrix $\mathbf{R}_i$.

Indeed, in this approach, the fuzzy set's description is almost exclusively relegated to the membership function, because, strictly speaking, its support is all of $\mathbb{R}^n$. Thus, the membership function is constructed around an underlying "support" set A. The set $A \subset \mathbb{R}^n$ under consideration is inscribed in an ellipsoid specified by the pair of parameters $(\bar{x}, \mathbf{R})$, where $\bar{x}$ is a vector in n-dimensional Euclidean space and $\mathbf{R}$ is a $n \times n$ real symmetric positive definite matrix. The following association is made:

$$A \equiv [x \mid (x - \bar{x})'\mathbf{R}^{-1}(x - \bar{x}) \leq c^2]$$

That is,

$$e^{\frac{-c^2}{2}} \leq \mu_A(x) \leq 1 \quad \forall\, x \in A$$

or in terms of "α-level set" terminology, $\alpha = e^{\frac{1}{2}c^2}$ for some constant c; without loss of generality, chose $c = 1$. Hence, the ellipsoidal set

$$A \equiv [x \mid (x - \bar{x})'\mathbf{R}^{-1}(x - \bar{x}) \leq 1] \tag{2.2}$$

is an approximation of the original set A that is of interest and, in fact, it directly determines the parameters $\bar{x}$ and $\mathbf{R}$. The latter parameterizes the Gaussian membership function $\mu_{A_i}(x)$. For example, a scalar $(A \subseteq \mathbb{R}^1)$ membership function

$$\mu_{\mathcal{A}}(x) = e^{-\frac{x^2}{2\sigma^2}}$$

is illustrated in Figure 2.1 for varying values of σ. Close examination of the figure indicates why the fuzzy set, say $\mathcal{A}$, could assume the linguistic description or the $\mathcal{A}$ label "x is zero". Indeed, the membership function $\mu_{\mathcal{A}}(x)$ from above assigns a membership value of 1 to $x = 0$ and, for $\mid x \mid \approx 0$, it assigns membership values close to 1. Furthermore, by choosing the parameter $\sigma \ll 1$, the meaning of "zero", or "x is small" is sharpened and, conversely, by choosing the σ parameter large, the
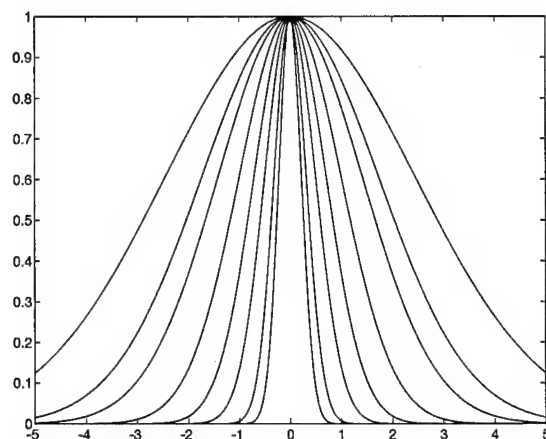
Figure 2.1   Gaussian Membership Function for Varying σ

meaning of "zero" or "x is small" is broadened and made quite inaccurate, so the verbal statements "x is zero" or "x is small" are then fuzzy. Moreover, one can set the "center of gravity" of the membership function μ_A at some prespecified value $\bar{x}$ of the variable x, viz.,

$$\mu_A(x) = e^{-\frac{(x-\bar{x})^2}{2\sigma^2}}$$

thereby giving a meaning to the fuzzy verbal statement "x is $\bar{x}$", or "x is near $\bar{x}$". Obviously, x close to $\bar{x}$, e.g., $x \ni \mid x - \bar{x} \mid \approx 0$, will be assigned by the membership function a value close to 1. Furthermore, by choosing the parameter σ small, only x very close to $\bar{x}$ will be assigned a high degree of membership in this set ($\mu(x) \approx 1$). Conversely, a large σ causes x relatively far away from $\bar{x}$ to be considered $\bar{x}$, by virtue of their membership function assigned value being pretty close to 1.

In general, the domain of definition of the membership function is the whole of $\mathbb{R}^n$, rather than an underlying support set A; however, from a practical point of view there is an underlying set A, implicitly specified by its "center" point (vector) $\bar{x} \in \mathbb{R}^n$ and by the square roots of the n eigenvalues of the "covariance" matrix $\mathbf{R}$, which determine its dimensions. Furthermore, similar to the scalar case, if all the eigenvalues of the $\mathbf{R}$ matrix are small, then the Gaussian membership function rapidly decreases to zero and in the limiting case of $|\mathbf{R}| \rightarrow 0$, one obtains the function

$$\mu_{\mathcal{A}}(x) = (2\pi)^{\frac{n}{2}} \sqrt{\det(\mathbf{R})} \, \delta(x - \bar{x}) \, ,$$

where $\delta(\cdot)$ is the multivariable "delta" function defined on $\mathbb{R}^n$ [34]. Hence, in this limiting case the fuzzy variable $\mathcal{A}$ is rendered a crisp (deterministic) variable which assumes the value $\bar{x}$.

If some eigenvalues of the $\mathbf{R}$ covariance matrix $\rightarrow 0$, this then indicates a crisp subspace in the x vector space, thus allowing for *mixed* fuzzy/crisp variables modeling. However, technically speaking, in the FLC calculations it is convenient to directly treat the fuzzy components, and at the same time momentarily consider the crisp state components to be *known* parameters. This is indeed a course of action that is adhered to in probability problems, where both random and deterministic variables are involved.

Finally, the universe of discourse can be further restricted by confining one's attention to a set $A_r \subset \mathbb{R}^n$, which then constitutes the domain of the above defined membership function.

For example, in the scalar case, the membership function is virtually 0 at "3σ", as is illustrated in Figure 2.1; hence, one can then say that the underlying set A is the segment $A = [-3\sigma, \, 3\sigma]$. In addition, the domain of definition of the membership

function can be chosen to correspond to a restricted universe of discourse of, say, $A_r = [0, \; 2\sigma]$.

In conclusion: A multivariable membership function approach is advocated. These weighing functions are modeled on the classical multivariate Gaussian probability density functions and their support is the whole of $\mathbb{R}^n$. Given an underlying set $A \subset \mathbb{R}^n$, the parameters of the corresponding Gaussian probability density functions are chosen, according to Eq. (2.2), to roughly model the state space region of interest in $\mathbb{R}^n$, in particular a region of the parameter space P. Hence, the weighing/membership function is relatively high there and it is very small outside this region - as required. The universe of discourse can be further delimited by specifying the domain A_r of the membership function.

2.2 Fuzzy Rules

Two types of linguistic "Rules" will be explored, for n states and m inputs:

1. Crisp (deterministic) output $\mathcal{R}_i$: **IF** x is $\mathcal{A}_i$, **THEN** apply a mapping. The mapping may be either

   state transition: The mapping $f_i : \; \mathbb{R}^n \to \mathbb{R}^n$, i.e., it is the function $f_i(x)$.

   or

   input: The mapping $g_i : \; \mathbb{R}^m \to \mathbb{R}^n$, i.e., it is the function $g_i(u)$.

2. Fuzzy (variable) output $\mathcal{R}_i$: **IF** x is $\mathcal{A}_i$, **THEN** the output y is $\mathcal{B}_i$.

In case 2, similar to the set $\mathcal{A}_i$ in the ith rule's antecedent, the fuzzy set $\mathcal{B}_i$ is defined by having recourse to a membership function $\nu_{\mathcal{B}_i}(y)$, where

$$\nu_{\mathcal{B}_i} : \; \mathbb{R}^n \; \to \; \mathbb{R}^+ \; ,$$

for the fuzzy state transition mapping, or

$$\nu_{\mathcal{B}_i} : \quad \mathbb{R}^m \quad \rightarrow \quad \mathbb{R}^+ \ ,$$

for the fuzzy input mapping.

The above membership functions are parameterized by $\bar{y}_i \in \mathbb{R}^n$ and the $n \times n$ real symmetric positive definite matrix $\mathbf{S}_i$ for the state transition map, or for the input map $\bar{y}_i \in \mathbb{R}^m$, in which case the real symmetric positive definite matrix $\mathbf{S}_i$ is $m \times m$. Hence, the membership function is

$$\nu_{\mathcal{B}_i}(y) = e^{-\frac{1}{2}(y - \bar{y}_i)'\mathbf{S}_i(y - \bar{y}_i)} \ .$$

This is as far as *fuzzification* goes.

2.3   Fuzzy Set Operations

In this research, the above outlined vector space approach is employed, where the fuzzy variables x or y represent points in Euclidean n dimensional or m dimensional vector spaces. Each component of the x or y vectors represents a particular fuzzy variable, but all the n or m fuzzy variables aggregated in the x or y vectors are jointly treated. Hence, the need to use composite clauses in the rules' antecedents, such as, $\mathcal{R}$: **IF** x_1 is "zero" **AND** x_2 is "positive medium" , **THEN** ... is obviated. Thus, the big advantage of the multivariable/vector space approach is that one need not, in some way, combine the antecedents' premises in order to calculate the $\mathcal{R}$ rule antecedent's strength, according to either the "Min rule"

$$\mu_{\mathcal{R}}(x_1, x_2) = \min(\mu_1(x_1), \ \mu_2(x_2))$$

or, alternatively, the sometimes preferred "product rule"

$$\mu_{\mathcal{R}}(x_1, x_2) = \mu_1(x_1) \, \mu_2(x_2)$$

In the currently proposed approach, one instead needs to judiciously set up the problem, according to the following modeling steps.

The procedure is first introduced for the special case of scalar fuzzy variables. Here, the fuzzy variables are $\mathcal{X}_1$, $\mathcal{X}_2$ and the scalar universes of discourse are $\mathsf{X}_1 \subseteq \mathbb{R}^1$, $\mathsf{X}_2 \subseteq \mathbb{R}^1$ and $x_1 \in \mathbb{R}^1$, $x_2 \in \mathbb{R}^1$.

1. *Universe of Discourse:* Form the cross product of the elementary, one dimensional, universes of discourse X_1 and X_2 and generate the universe of discourse $\mathsf{X} = \mathsf{X}_1 \times \mathsf{X}_2 \subseteq \mathbb{R}^2$. Thus, if $x_1 \in \mathsf{X}_1$ and $x_2 \in \mathsf{X}_2$, then $x = (x_1, x_2) \in \mathsf{X}$.

2. *Fuzzification example:* Let $\bar{x}_2 > 0$ represent what is considered to be a medium sized value of the variable x_2. The x_2 values of the variable $\mathcal{X}_2$, that are within a distance $3\sigma_2$ of the "benchmark" $\bar{x}_2$, are considered "positive medium". Also, in the spirit of fuzzy logic, consider values of x_1 that are in absolute value less than $3\sigma_1$, to virtually be "zero". Hence, the membership functions of the fuzzy variables "$\mathcal{X}_1$ is zero" and "$\mathcal{X}_2$ is positive medium" are

$$\begin{aligned}
\mu_{\mathcal{X}_1}(x_1) &= e^{-\frac{1}{2}\frac{x_1^2}{\sigma_1^2}} \\
\mu_{\mathcal{X}_2}(x_2) &= e^{-\frac{1}{2}\frac{(x_2-\bar{x}_2)^2}{\sigma_2^2}}
\end{aligned}$$

respectively.

3. *Logical "AND" injunction operation:* Create the membership function

$$\mu_{\mathcal{R}}(x) = \mu_{(\mathcal{X}_1 \text{ AND } \mathcal{X}_2)}(x) = e^{-\frac{1}{2}\left[\frac{x_1^2}{\sigma_1^2}+\frac{(x_2-\bar{x}_2)^2}{\sigma_2^2}\right]}$$
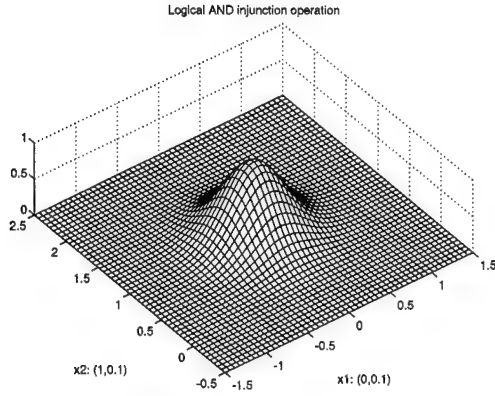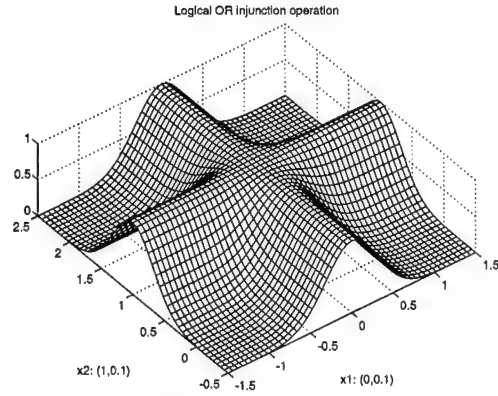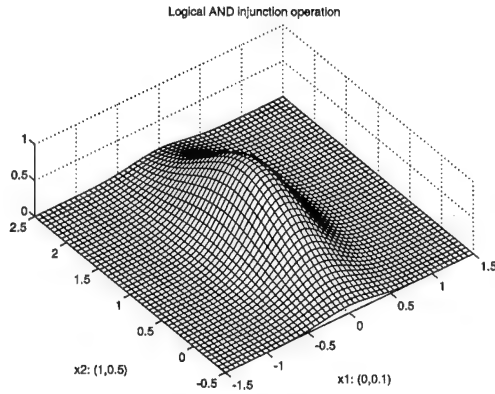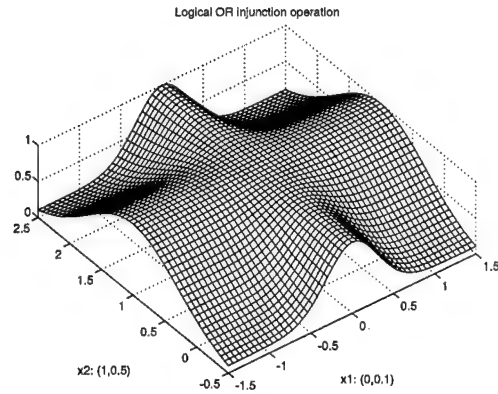
Figure 2.2   Surface Plots of Bivariate Fuzzy Logical Injunction Operators

where $x = (x_1, x_2)$. The above membership function possesses the desired attributes, for it penalizes deviations away from both $x_1 = 0$ and $x_2 = \bar{x}_2$. Furthermore, in view of the well known properties of the exponential function, it is now evident that the approach constitutes a generalization of the above mentioned "product rule" for the "AND" injunction. The construction is illustrated in Figure 2.2a for $\bar{x}_2 = 1$, $\sigma_1^2 = \sigma_2^2 = 0.1$ and in Figure 2.2c for $\sigma_1^2 = 0.1$, $\sigma_2^2 = 0.5$.

4. *Logical "OR" injunction operation:* If in the $\mathcal{R}$ rule example from above the "AND" logical injunction is replaced by the "OR" logical injunction, the strength of the rule's antecedent is obtained as follows.

Consider Figure 2.2b. The $\mu_{(\mathcal{X}_1 \text{ OR } \mathcal{X}_2)}$ membership function should be relatively high in the cruciform - like region, which is the union of the two strips in the Euclidean plane that represent the respective support sets $\bar{x}_1 - \sigma_1 \leq x_1 \leq \bar{x}_1 + \sigma_1$ and $\bar{x}_2 - \sigma_2 \leq x_2 \leq \bar{x}_2 + \sigma_2$. The ensuing cross - like region is not convex. The fuzzy union could be obtained using the "Max" operator, but this does not yield a smooth transition. It is required that the membership function be relatively large inside the above mentioned region, and for it to be small outside the region. Hence, to obtain a smooth result let

$$\mu_{(\mathcal{X}_1 \text{ OR } \mathcal{X}_2)}(x) = \mu_{\mathcal{X}_1}(x_1) + \mu_{\mathcal{X}_2}(x_2) - \mu_{(\mathcal{X}_1 \text{ AND } \mathcal{X}_2)}(x)$$

where the $\mu_{(\mathcal{X}_1 \text{ AND } \mathcal{X}_2)}$ membership function has been constructed according to 2 above and $x = (x_1, x_2)$. Therefore,

$$\mu_{(\mathcal{X}_1 \text{ OR } \mathcal{X}_2)}(x_1, x_2) = e^{-\frac{1}{2}\frac{x_1^2}{\sigma_1^2}} + e^{-\frac{1}{2}\frac{(x_2 - \bar{x}_2)^2}{\sigma_2^2}} - e^{-\frac{1}{2}\left[\frac{x_1^2}{\sigma_1^2} + \frac{(x_2 - \bar{x}_2)^2}{\sigma_2^2}\right]}$$

this is in fact the functions depicted in Figures 2.2b,d.

5. *Logical "NOT" injunction operation:* Is shown in Figure 2.3 for $\bar{x}_2 = 1$, $\sigma_1^2 = \sigma_2^2 = 0.1$ when defined as

$$\mu_{(\text{NOT } \mathcal{X})}(x) = 1 - \mu_{\mathcal{X}}(x)$$

It transpires from Step 3 in the above discussion how, from given scalar fuzzy variables, multidimensional fuzzy variables are being built up.

a) NOT $\mathcal{X}_1$ b) NOT $\mathcal{X}_2$
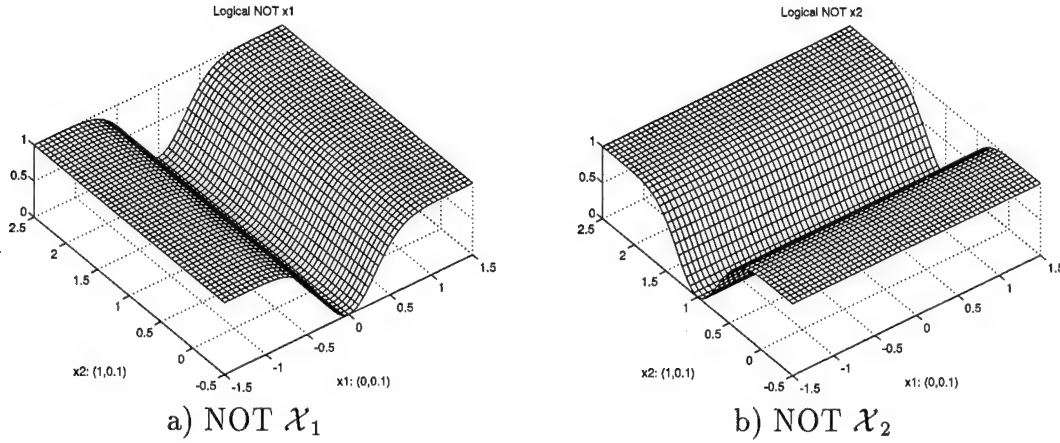
Figure 2.3   Surface Plots of Fuzzy Logical NOT Injunction Operator

In the general case, the multidimensional fuzzy sets $\mathcal{X}_1$ and $\mathcal{X}_2$ are considered. The "Universes of Discourse" are then $X_1 \subseteq \mathbb{R}^{n_1}$ and $X_2 \subseteq \mathbb{R}^{n_2}$; $n_1 \geq 1$, $n_2 \geq 1$. The general modeling procedure is given in the sequel.

1. Universe of Discourse:

$$X = X_1 \times X_2 \subseteq \mathbb{R}^{n_1 + n_2}$$

with $x = (x_1, x_2)$, where $x_1 \in X_1$, $x_2 \in X_2$.

2. Fuzzification: Let the "representative" vectors be $\bar{x}_1 \in \mathbb{R}^{n_1}$, $\bar{x}_2 \in \mathbb{R}^{n_2}$, and where $\mathbf{R}_1$ and $\mathbf{R}_2$ are real symmetric $n_1 \times n_1$ and $n_2 \times n_2$ positive definite matrices, respectively. Thus, the membership functions are

$$\mu_{\mathcal{X}_1}(x_1) = e^{-\frac{1}{2}(x_1 - \bar{x}_1)' \mathbf{R}_1^{-1}(x_2 - \bar{x}_2)}$$

$$\mu_{\mathcal{X}_2}(x_2) = e^{-\frac{1}{2}(x_2 - \bar{x}_2)' \mathbf{R}_2^{-1}(x_2 - \bar{x}_2)}$$

3. Logical "AND" injunction operation:

$$\mu_{\mathcal{R}}(x) = \mu_{(\mathcal{X}_1 \text{ AND } \mathcal{X}_2)}(x) = e^{-\frac{1}{2}[(x_1-\bar{x}_1)'\mathbf{R}_1^{-1}(x_1-\bar{x}_1)+(x_2-\bar{x}_2)'\mathbf{R}_2^{-1}(x_2-\bar{x}_2)]} \quad (2.3)$$

where $x = (x_1, x_2)$.

4. Logical "OR" injunction operation:

$$
\begin{aligned}
\mu_{\mathcal{R}}(x) = \mu_{(\mathcal{X}_1 \text{ OR } \mathcal{X}_2)}(x) &= \mu_{\mathcal{X}_1}(x_1) + \mu_{\mathcal{X}_2}(x_2) - \mu_{(\mathcal{X}_1 \text{ AND } \mathcal{X}_2)}(x) \\
&= e^{-\frac{1}{2}(x_1-\bar{x}_1)'\mathbf{R}_1^{-1}(x_1-\bar{x}_1)} + e^{-\frac{1}{2}(x_2-\bar{x}_2)'\mathbf{R}_2^{-1}(x_2-\bar{x}_2)} \\
&\quad - e^{-\frac{1}{2}[(x_1-\bar{x}_1)'\mathbf{R}_1^{-1}(x_1-\bar{x}_1)+(x_2-\bar{x}_2)'\mathbf{R}_2^{-1}(x_2-\bar{x}_2)]} \quad (2.4)
\end{aligned}
$$

where $x = (x_1, x_2)$.

5. Logical "NOT" injunction operation: Given a fuzzy variable $\mathcal{X}$, the fuzzy variable "NOT $\mathcal{X}$" is specified by its membership function

$$\mu_{(\text{NOT } \mathcal{X})}(x) = 1 - \mu_{\mathcal{X}}(x) = 1 - e^{-\frac{1}{2}(x-\bar{x})'\mathbf{R}^{-1}(x-\bar{x})} \quad (2.5)$$

where $x \in \mathsf{X} \subseteq \mathbb{R}^{n_1+n_2}$.

Theorem 1 *The above constructed function $\mu_{(\mathcal{X}_1 \text{ OR } \mathcal{X}_2)}(x) : \mathbb{R}^{n_1+n_2} \to \mathbb{R}^1$ is indeed a "membership function". That is,*

1. It is non-negative, i.e.,

$$\mu_{(\mathcal{X}_1 \text{ OR } \mathcal{X}_2)}(x_1, x_2) > 0 \quad \forall x_1 \in \mathbb{R}^{n_1}, x_2 \in \mathbb{R}^{n_2}$$

2. It is bounded above:

$$\mu_{(\mathcal{X}_1 \text{ OR } \mathcal{X}_2)}(x_1, x_2) \le 1 \quad \forall x_1 \in \mathbb{R}^{n_1}, x_2 \in \mathbb{R}^{n_2}$$

3. It is normalized, i.e.,

$$\max_{x \in \mathsf{X}} \ \mu_{(\mathcal{X}_1 \text{ OR } \mathcal{X}_2)}\left(x_1, x_2\right) = \mu_{(\mathcal{X}_1 \text{ OR } \mathcal{X}_2)}\left(\bar{x}_1, \bar{x}_2\right) = 1$$

4. It assumes relatively large values for $x \in \mathsf{A}$, where A is a nonconvex cruciform set

$$\mathsf{A} = [x \mid x = (x_1, x_2), \ (x_1 - \bar{x}_1)' \mathbf{R}_1^{-1}(x_1 - \bar{x}_1) \le 1, \ (x_2 - \bar{x}_2)' \mathbf{R}_2^{-1}(x_2 - \bar{x}_2) \le 1]$$

and for $x \notin \mathsf{A}$, the membership function "vanishes".

Proof

<u>For 1:</u> By the definition of the "OR" operator

$$
\begin{aligned}
\mu_{(\mathcal{X}_1 \text{ OR } \mathcal{X}_2)}(x_1, x_2) &= \mu_{\mathcal{X}_1}(x_1) + \mu_{\mathcal{X}_2}(x_2) - \mu_{\mathcal{X}_1}(x_1)\mu_{\mathcal{X}_2}(x_2) & (2.6) \\
&= \mu_{\mathcal{X}_1}(x_1)[1 - \mu_{\mathcal{X}_2}(x_2)] + \mu_{\mathcal{X}_2}(x_2) & (2.7) \\
&\ge 0
\end{aligned}
$$

Since

$$\mu_{\mathcal{X}_2}\left(x_2\right) \le 1$$

and the Gaussian multivariate distributions $\mu_{\mathcal{X}_1}(x_1)$ and $\mu_{\mathcal{X}_2}(x_2)$ are non-negative.

<u>For 2:</u> Application of the triangle inequality to Eq. (2.6), using the fact that Gaussian multivariate distributions are non-negative yields.

$$
\begin{aligned}
\left|\mu_{(\mathcal{X}_1 \text{ OR } \mathcal{X}_2)}(x_1, x_2)\right| &\le \left|\mu_{\mathcal{X}_1}(x_1)\right| + \left|\mu_{\mathcal{X}_2}(x_2)\right| - \left|\mu_{\mathcal{X}_1}(x_1)\right|\left|\mu_{\mathcal{X}_2}(x_2)\right| \quad \forall \ x_1 \in \mathbb{R}^{n_1}, \ x_2 \in \mathbb{R}^{n_2} \\
&\le 1 + 1 - 1 \\
&\le 1
\end{aligned}
$$

2-13

which, along with 1 above, implies 2.

For 3: Calculate the maximum of the function $\mu_{(\mathcal{X}_1 \text{ OR } \mathcal{X}_2)}(x_1, x_2)$ on $\mathbb{R}^{n_1+n_2}$. To this end, set

$$\frac{\partial \mu_{(\mathcal{X}_1 \text{ OR } \mathcal{X}_2)}(x_1, x_2)}{\partial x_1} = 0$$

Eq. (2.7) thus yields

$$[1 - \mu_{\mathcal{X}_2}(x_2)]\frac{\partial \mu_{\mathcal{X}_1}(x_1)}{\partial x_1} = -[1 - \mu_{\mathcal{X}_2}(x_2)]\left[\mathbf{R}_1^{-1}(x_1 - \bar{x}_1)\right]\mu_{\mathcal{X}_1}(x_1) = 0$$

The above equation implies that an extremum point may be attained if either $x_1 = \bar{x}_1$, or $\mu_{\mathcal{X}_2}(x_2) = 1$ which occurs at $x_2 = \bar{x}_2$. A similar conclusion is reached if the partial with respect to x_2 of the above function is set to zero. Hence, $\bar{x}_1$ and $\bar{x}_2$ constitute an extremal point, where a candidate local maximum may be attained. Inserting $x_1 = \bar{x}_1$ and $x_2 = \bar{x}_2$ into Eq. (2.7) above gives the value of unity. Thus $(\bar{x}_1, \bar{x}_2)$ is a global maximum, by 2 above, and part 3 is proven.

For 4: This follows directly from the analytic properties of the multivariate Gaussian distribution.

∎

In conclusion, the modeling approach presented in steps 1-5 has the following advantages.

- It is analytic, in contrast to the non-smooth membership functions that ensue when the Max and Min operators are invoked for the "OR" and "AND" logical injunctions, respectively.

- The proposed modeling approach is in the true spirit of Probability Theory.

2.4 More Fuzzy Logic

In the same vein, the following additional issues are addressed.

2.4.1 Conflict Resolution. Consider a set of N crisp (type 1) rules. The need for "Conflict Resolution" arises in the instance where more than one, say N, Fuzzy Logic Control rule fires at one time. It is treated as follows. The fuzzified mapping from $\mathbb{R}^n$ to $\mathbb{R}^+$ then is

$$f(x) = \frac{\sum_{i=1}^{N} \mu_{\mathcal{A}_i}(x) \, f_i(x)}{\sum_{i=1}^{N} \mu_{\mathcal{A}_i}(x)}$$

REMARK: If the mappings $f_i(x) = g(x) \; \forall \, i = 1, 2, \dots, N$, then the fuzzified mapping $f(x) = g(x)$.

Hence, the fuzzified mapping is explicitly given by

$$f(x) = \frac{1}{\sum_{i=1}^{N} e^{-\frac{1}{2}(x-\bar{x}_i)'\mathbf{R}_i^{-1}(x-\bar{x}_i)}} \sum_{i=1}^{N} e^{-\frac{1}{2}(x-\bar{x}_i)'\mathbf{R}_i^{-1}(x-\bar{x}_i)} f_i(x) \qquad \forall \, x \in \mathbb{R}^n \qquad (2.8)$$

In the special case where the fuzzy variables are "similar", viz., $\mathbf{R}_i = \mathbf{R}$ for every $i = 1, 2, \dots, N$, the above formula is

$$f(x) = \frac{1}{\sum_{i=1}^{N} e^{\bar{x}_i'\mathbf{R}^{-1}(x-\frac{1}{2}\bar{x}_i)}} \sum_{i=1}^{N} e^{\bar{x}_i'\mathbf{R}^{-1}(x-\frac{1}{2}\bar{x}_i)} f_i(x)$$

Also, note that:

- All the N rules come into play, $\forall \, x \in \mathsf{A}_r$, i.e., the domain of the ensuing fuzzified mapping $f(x)$ is the whole universe of discourse.

- $f(x)$ is analytic.

Example: x is a scalar and the universe of discourse is the set $\mathsf{A}_r = [-2, \, 2]$. The membership functions, which characterize the fuzzy variables $\mathcal{X}_1$ "x is Negative", $\mathcal{X}_2$ "x is Zero" and $\mathcal{X}_3$ "x is Positive" have the common universe of discourse A_r and are parameterized by $\bar{x}_i, \sigma_i, i = 1, 2, 3$, respectively, where $\bar{x}_1 = -2, \bar{x}_2 = 0, \bar{x}_3 = 2$ and $\sigma_1 = \sigma_2 = \sigma_3 = 1$. These membership functions are illustrated in Figure 2.4.
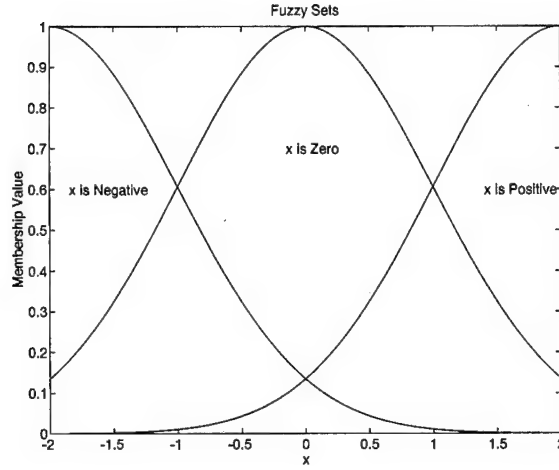
Figure 2.4   Fuzzy Variables "x is Negative", "x is Zero", and "x is Positive"

Next, consider the following three fuzzy rules which verbally describe a fuzzy state transition mapping $f(x)$ of a hard saturation element; these linguistic rules are:

$\mathcal{R}_1$: **IF** x is "Negative" **THEN** $f_1(x) = -1$.

$\mathcal{R}_2$: **IF** x is "Zero" **THEN** $f_2(x) = x$.

$\mathcal{R}_3$: **IF** x is "Positive" **THEN** $f_3(x) = +1$.

The ensuing fuzzified state transition mapping

$$
\begin{aligned}
f(x) &= \frac{-e^{-\frac{1}{2}(x+2)^2} + xe^{-\frac{1}{2}x^2} + e^{-\frac{1}{2}(x-2)^2}}{e^{-\frac{1}{2}(x+2)^2} + e^{-\frac{1}{2}x^2} + e^{-\frac{1}{2}(x-2)^2}} \\
&= \frac{-e^{-2(1+x)} + x + e^{-2(1-x)}}{e^{-2(1+x)} + 1 + e^{-2(1-x)}} \\
&= \frac{x + 2e^{-2}\sinh(2x)}{1 + 2e^{-2}\cosh(2x)}.
\end{aligned}
$$

is graphically depicted in Figure 2.5a.

Additional example: Consider the fuzzy state transition mapping:
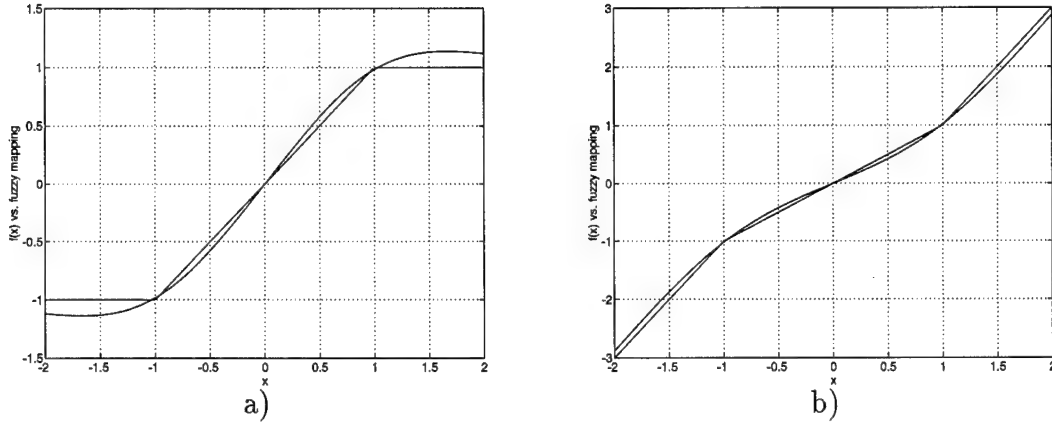
Figure 2.5   Fuzzified Mapping of Continuous Nonlinear Elements

$\mathcal{R}_1$: **IF** x is "Negative" **THEN** $f_1(x) = 2x + 1$.

$\mathcal{R}_2$: **IF** x is "Zero" **THEN** $f_2(x) = x$.

$\mathcal{R}_3$: **IF** x is "Positive" **THEN** $f_3(x) = 2x - 1$.

The ensuing fuzzified state transition mapping is

$$
\begin{aligned}
f(x) &= \frac{(2x+1)e^{-\frac{1}{2}(x+2)^2} + xe^{-\frac{1}{2}x^2} + (2x-1)e^{-\frac{1}{2}(x-2)^2}}{e^{-\frac{1}{2}(x+2)^2} + e^{-\frac{1}{2}x^2} + e^{-\frac{1}{2}(x-2)^2}} \\
&= \frac{x[1 + 4e^{-2}\cosh(2x)] - 2e^{-2}\sinh(2x)}{1 + 2e^{-2}\cosh(2x)}
\end{aligned}
$$

and it is graphically depicted in Figure 2.5b.

In Figure 2.5, the ensuing nonlinear fuzzy maps are plotted alongside the underlying original piece-wise linear maps that were featured in the fuzzy rules. The fuzzy maps are smooth, viz., they are, by construction, analytic, and the fit is remarkably good.
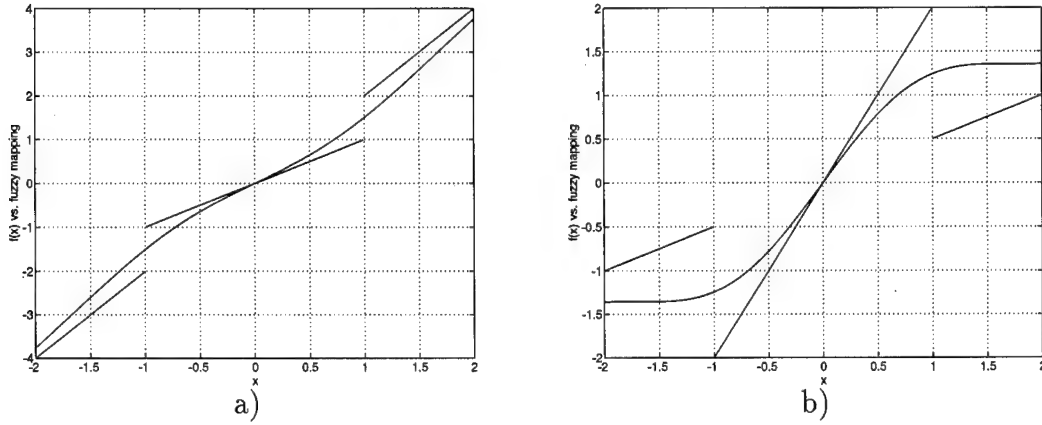
Figure 2.6   Fuzzified Mapping of Discontinuous Nonlinear Elements

In the above examples, the original map featured in the fuzzy rules is "continuous". Two additional examples, where the map in the fuzzy rules is "discontinuous", are presented in Figure 2.6. In Figure 2.6a, the fuzzy state transition mapping is:

$\mathcal{R}_1$: **IF** x is "Negative" **THEN** $f_1(x) = 2x$.

$\mathcal{R}_2$: **IF** x is "Zero" **THEN** $f_2(x) = x$.

$\mathcal{R}_3$: **IF** x is "Positive" **THEN** $f_3(x) = 2x$.

whose fuzzified state transition mapping is expressed by

$$f(x) = \left(2 - \frac{1}{1 + 2e^{-2}\cosh{(2x)}}\right)x$$

In Figure 2.6b, the fuzzy state transition mapping is:

$\mathcal{R}_1$: **IF** x is "Negative" **THEN** $f_1(x) = \frac{1}{2}x$.

$\mathcal{R}_2$: **IF** x is "Zero" **THEN** $f_2(x) = 2x$.

$\mathcal{R}_3$: **IF** x is "Positive" **THEN** $f_3(x) = \frac{1}{2}x$.

2-18

whose fuzzified state transition mapping is expressed by

$$f(x) = \frac{\frac{1}{2}xe^{-\frac{1}{2}(x+2)^2} + 2xe^{-\frac{1}{2}x^2} + \frac{1}{2}xe^{-\frac{1}{2}(x-2)^2}}{e^{-\frac{1}{2}(x+2)^2} + e^{-\frac{1}{2}x^2} + e^{-\frac{1}{2}(x-2)^2}}$$

$$= \frac{1}{2}x \frac{4 + 2e^{-2}\cosh(2x)}{1 + 2e^{-2}\cosh(2x)}$$

2.4.2 Defuzzification. For fuzzy rules of type 2 above, the requirement of a crisp (deterministic) output variable entails a "defuzzification" step. Thus, assume that the fuzzy variables $\mathcal{B}_i$ $i = 1, 2, \ldots, N$ have a *common* universe of discourse and are parameterized by the set of N pairs $(\bar{y}_i, \mathbf{S}_i)$, where $\bar{y}_i \in \mathbb{R}^m$ and the real symmetric positive definite matrices $\mathbf{S}_i$ are $m \times m$. A "Maximum Likelihood" formulation is proposed, where the output

$$y = \frac{\sum_{i=1}^{N} \mu_{\mathcal{A}_i}(x)\,\bar{y}_i}{\sum_{i=1}^{N} \mu_{\mathcal{A}_i}(x)}$$

Hence

$$y(x) = \frac{1}{\sum_{i=1}^{N} e^{-\frac{1}{2}(x-\bar{x}_i)'\mathbf{R}_i^{-1}(x-\bar{x}_i)}} \sum_{i=1}^{N} e^{-\frac{1}{2}(x-\bar{x}_i)'\mathbf{R}_i^{-1}(x-\bar{x}_i)}\bar{y}_i \qquad (2.9)$$

In the special case where all input variables are "similar", viz., $\mathbf{R}_i = \mathbf{R}$ for every $i = 1, 2, \ldots, N$, the above formula simplifies to

$$y(x) = \frac{1}{\sum_{i=1}^{N} e^{\bar{x}_i'\mathbf{R}^{-1}(x-\frac{1}{2}\bar{x}_i)}} \sum_{i=1}^{N} e^{\bar{x}_i'\mathbf{R}^{-1}(x-\frac{1}{2}\bar{x}_i)}\bar{y}_i$$

In the current formulation, the input/output mapping

$$y(x) \ : \ \mathbb{R}^n \rightarrow \mathbb{R}^m$$

which is given by the above closed - form Eq. (2.9), is defined for all $x \in \mathsf{A}_r$ and it is analytic.

In the special case where the (common) universe of discourse is $\mathbb{R}^m$, the above formula reduces to the often used Center Of Area (COA) rule, for then $\bar{y}_i$ is the center point of the fuzzy set $\mathcal{B}_i$, $i = 1, 2, \ldots, N$.

III. The n-Dimensional Scheduler

The net product of this research is the development of a control synthesis method applicable to nonlinear systems with structured parametric uncertainty. This controller acknowledges the sensitivity of the plant's model to the operating point within the entire envelope of the system. The solution is obtained by scheduling several parallel point controllers (which may be robust with specified operating regions), each independently designed, on measurements of both fast internal states and varying parameters on which the plant's model is dependent. It is the scheduling on the fast states as opposed to only 'slow' states or scheduling parameters (e.g., dynamic pressure $\bar{q}$ in flight control), and uncertain parameters, which allows for the direct handling of system nonlinearities. The developed synthesis technique of this research not only gives the means of scheduling the point designs during operation, but also the actual location of the point designs themselves, yielding the total solution of the controller.

The resulting controller is for multiple-input single-output (MISO) systems. The dimension of the input for the scheduler is that of the plant under control, plus additional elements of a parameter vector. As such, the technique can be used in conjunction with multiple-input multiple-output (MIMO) controller design techniques whose solutions requires solving several MISO problems, thus bridging the gap to full envelope control of MIMO nonlinear systems with parametric variation. This is accomplished without the need for trial and error in the placement of point controllers or their respective membership functions.

Let p denote a vector whose n elements are comprised of both the internal states and the varying model parameters which are to be scheduled upon. Each element of p has a range of admissible values dictated by the desired operational envelope of the controlled system. The admissible domain of p then defines the parameter space, $\mathsf{P} \subseteq \mathbb{R}^n$, or the domain of the system over which the point controllers must be

scheduled. For a fixed $p \in$ P the plant of interest is represented by a deterministic nonlinear set of equations. Conventional LTI theory can be used to design a point controller for a linear model obtained by linearization about this operating point. However, the controller's performance is only "as designed" in a yet to be determined "region of attraction" about this operating point. Or in the case of robust controllers, this region may be known but it is not large enough to encompass all of P. Since the objective is to achieve a specified level of performance over the entire envelope, the size of these individual valid control regions effects the total number, as well as placement, of "point" controllers that will be required.

Denote a set of points in P as G $= \{p_i\}_{i=1}^{N}$. Given a set of N point controllers, each designed about $p_i \in$ G, a scheduling scheme is based upon multivariate Fuzzy Logic as developed in Chapter II. The control authority of each controller is the normalized membership value of the current operating point within each controller's corresponding membership function. That is, the implementation of the fuzzy scheduling block of Figure 1.1 on page 1-6, while operating at $p \in$ P $\subseteq \mathbb{R}^n$, is performed by scaling each controller's output by $W_i(p)$ where

$$W_i(p) = \frac{\mu_i(p)}{\sum_{j=1}^{N} \mu_j(p)} \tag{3.1}$$

The experiments contained in Appendix B provide a basis for the selection of the above controller structure. They also highlight the need to quantify the two *coupled* problems of point controller location and membership function parameter selection, to be addressed in the sequel.

This research addresses the question of placement and number of point controllers by means of an optimization scheme with constraints on the controlled system's performance. The optimization is posed such that the solution of the Kuhn-Tucker equations [27, 28] yield the desired result. This is achieved by use of a Sequential Quadratic Programming (SQP) routine in the examples to follow, though

the problem is posed to allow choice of an optimization algorithm. The multivariate membership functions are chosen as Gaussian, each centered at $p_i \in G$ with variance parameter matrices $\mathbf{R}_i$ obtained from an optimization (derived in Section 3.4) based upon the location of all the other point designs. In the case of robust point designs, p_i can be viewed as the center of mass of the controller's specified operating region in P.

3.1   Location of Point Controllers

The objective is to design a full envelope controller for a given system. This controller is to provide not only stable, but "acceptable" tracking performance, for step commands over the entire operational envelope, P, of the system. Being able to slew the nonlinear system is a nontrivial task and a primary objective. To guide the design process, an optimality condition should be chosen which yields the desired results.

One could choose to try for the "best" response, where in order to yield an implementable solution the number of allowable point designs must be constrained. This goal has two problems: 1) identifying the best, and 2) constraining the number of point designs a priori. Defining a "best" response can be very subjective in many tasks. In a much larger class of problems, a response can be judged as good enough or no better/worse than another response in a more objective fashion. The number of design points required is discussed in the sequel, for now assume a sufficient number exist. Instead of the best response, a more useful preliminary performance optimization criterion is chosen as:

Given the operational envelope, P, find the minimum number of point designs which yield "acceptable" responses over P.

The constraint is now in the form of a functional operating on the output response. The resulting point controllers are now guaranteed to provide sufficient cover by means of meeting the performance constraint. There now exists a means

3-3

of quantifying sufficient cover, and it is a direct byproduct of the optimization. However, since the topology of the optimal cover is part of the solution, it is not known a priori.

To quantify the number of point designs required, in the optimization scheme to follow, one assumes a priori that the number chosen can provide sufficient cover of the operational envelope P. If the assumption proves false, one merely increases the number and obtains a new solution. Hence, rather then directly solving the above stated preliminary optimization, it is chosen to solve the dual problem of:

Maximize the coverage of the scheduler, given a fixed number of available point controllers, such that "acceptable" responses are achieved.

This problem statement requires specifying the number of controllers, a priori, but the optimization guarantees maximal cover over the resulting region. For this fixed number of controllers, $N \in \mathbb{N}$, if the scheduler's coverage exceeds P then N may be reduced. If the scheduler does not cover the full envelope, N must be increased. At this time a quantitative specification of cover is required, and it is taken as the volume of the convex hull ($\subset \mathbb{R}^n$) generated by G. Denote this map as $H : \mathsf{G} \to \mathbb{R}^+$. For the n-dimensional scheduling problem at hand, p_i is the point at which the ith controller is designed. Then, for a fixed N, the optimization problem is:

$$\max_{p_i \in \mathsf{P}} \ H(\mathsf{G}) \qquad (3.2)$$

$$\text{such that} \qquad (3.3)$$

$$C(y(t)) = 0 \qquad (3.4)$$

where $y(t)$ is the system response and C is the yet to be defined constraint functional. An equality constraint is used since the constraint functional will be developed similar to a cost function which will not penalize acceptable responses but yield positive cost for unacceptable responses.

At this time it is necessary to quantify the meaning of full envelope control with respect to the response constraint for LTI and non-LTI plants. There are two requirements of the controller: stability and transient response for tracking commands. In the case of an LTI plant, one is only scheduling on parameter uncertainty, say for instance τ, and this is how scheduling is classically used. This is an easier task than scheduling for model uncertainty, such as using the LTI approximation of a non-LTI plant, since it is more structured and is fully modeled by the parametric representation of the LTI plant. For a given τ, one has a deterministic LTI closed loop system and stability is determined by its eigenvalues. All bounded step commands are admissible, and in fact their responses are identical if normalized so only unity steps need be used. The stability and transient performance are independent of the initial trim point, and therefore are started from zero. This is not the case for Linear Time-Varying (LTV) and nonlinear plants. The initial condition and command strength, as well as sign of the command all effect the stability and transient performance. The following clarifies the interpretation of full envelope control for non-LTI systems.

Given that the system starts at rest from some trim condition, $x \in \mathsf{P}$, define the set of admissible step commands, S_x, as those which take the system from x to another point $z \in \mathsf{P}$ such that the convex combination of the x and z remains in P. That is,

$$\mathsf{S}_x = \{z - x \mid \lambda z - (\lambda - 1)x \ \in \ \mathsf{P}, \quad \forall\, \lambda \in [0,1]\} \tag{3.5}$$

When P is convex,

$$\mathsf{S}_x = \{z - x \mid z \in \mathsf{P}\} \tag{3.6}$$

Define the action of the controlled system on the input $s \in \mathsf{S}_x$ from trim condition $x \in \mathsf{P}$ as $\Phi(s|x) = y(t)$, where $y(t)$ is the output response. For stability,

one requires $\forall\, x \in \mathsf{P}$ and every s in its corresponding S_x that

$$\|\Phi(s|x) - z\| < \epsilon \in \mathbb{R}^+ \text{ as } t \to \infty \tag{3.7}$$

where ϵ represents a bounded maximum steady-state tracking error.

Next, the transient response is considered, and it is quantified by the functional C to be developed in the sequel. One could apply the transient constraints on all admissible commanded inputs. However, the intent is to allow the designer to use linear control tools at the design points, and to be able to design the point controllers independent of each other. Allowing this amount of design freedom and constraining the responses of all admissible commands may over constrain the optimization until no solution exists. A more appropriate scheme is to apply the transient constraints to some subset of S_x. In this work, a slewing between point designs paradigm is chosen. That is, the transient constraint is applied only to responses commanded to points in the region of P bounded by the closest point designs in every direction. So, the solution guarantees transient response as it slews between point designs and stability is checked against all admissible commands, S_x.

3.2 Nearest Neighbor in n-Dimensions

In the above definition of transient constraints, it is required to determine the *nearest neighbors* of a point in n-dimensions. One might choose to interpret this by means of orthogonal projections of $p - p_i \in \mathsf{P}$. Although this may be appropriate if one assumes a uniform spacing of the point controllers in P, this assumption is a poor one. The placement of the point controllers is the outcome of a constrained optimization based upon the dynamics of the plant. One innovation of the developed technique is that the resulting point controller locations reflect the model's dynamics. The points are more spread out in regions where the model is insensitive to changes in operating point, and the points are closer together in regions where the model

is more sensitive to such changes. Thus, the spacing reflects the size and shape of the regions about each point that a linearized model is valid. Uniform placement of plants is indicative of a nearly linear system.

Instead, and following the *computational geometry* paradigm, a better definition is used. Define the set of nearest neighbors of point p_j as $N_j \subseteq G$. For point $p_j \in G$, form the bisecting hyperplanes B_i between p_j and $p_i \in G$, $i \neq j$. Then form the hypercell $V_j \subset \mathbb{R}^n$ as

$$V_j = \bigcap_{i=1}^{N} B_i, \quad i \neq j \tag{3.8}$$

Points in the interior of this cell are closer to p_j than to any other element of G. The resulting boundary faces of this cell are subsets of the bisecting hyperplanes of nearest neighbors. That is,

$$p_i \in N_j \iff B_i \bigcap V_j \neq \emptyset \tag{3.9}$$

The cell that is formed and the neighbors it defines is of great use in the general n-dimensional scheduler that is developed.

The constructed cells have been used in many fields of Applied Mathematics, dating back to 1908, and are known as Voronoi diagrams or Thiessan Polygons [9]. This research introduces the Voronoi diagram to control, and, in particular, the field of n-dimensional scheduling of controllers, *a natural extension of its application.* Since the impetus for its consideration is to find generalized neighbors of points, its geometric dual, the Delaunay triangulation, is used instead [9]. The Delaunay diagram can be viewed as the unique triangulation which connects each point of a set with all of its neighbors as generated from the Voronoi diagram. Examples of these diagrams are depicted in Figure 3.1 for an arbitrary set of points in $\mathbb{R}^2$.

a) Voronoi Diagram   b) Delaunay Diagram
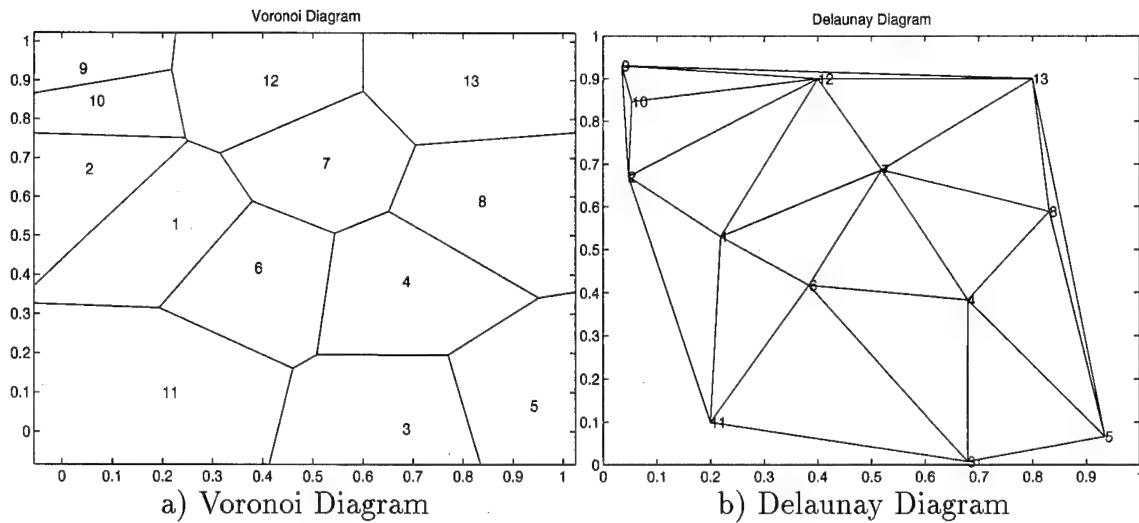
Figure 3.1   Generalized Neighbors

Generation of these constructs on a finite precision computer is a problem of current interest in Computational Geometry. The approach taken in this research is a composite of many suggested techniques [9, 14, 37, 24] from the current literature. The set of MATLAB® functions to generate these constructs are contained in Appendix C. The problem can also be posed for solution by neural networks where the appropriate geometry can be extracted by Topology Representing Networks (TRNs) [31, 41]. The TRN approach becomes increasingly attractive as the dimension of the parameter space increases.

3.3   Development of the Constraint Functional

As mentioned earlier, the goal of the constraint functional is to quantify whether or not the system's transient response is "acceptable". It is therefore used to validate sufficient cover between point designs. Classically this is performed by inspection of the transients or one of two quantification methods; output error from some reference or remaining within acceptable transient thumb prints. While these methods have their merits, all have major shortfalls. Visual inspection does not lend itself to automated and autonomous (with no human intervention) techniques and hence
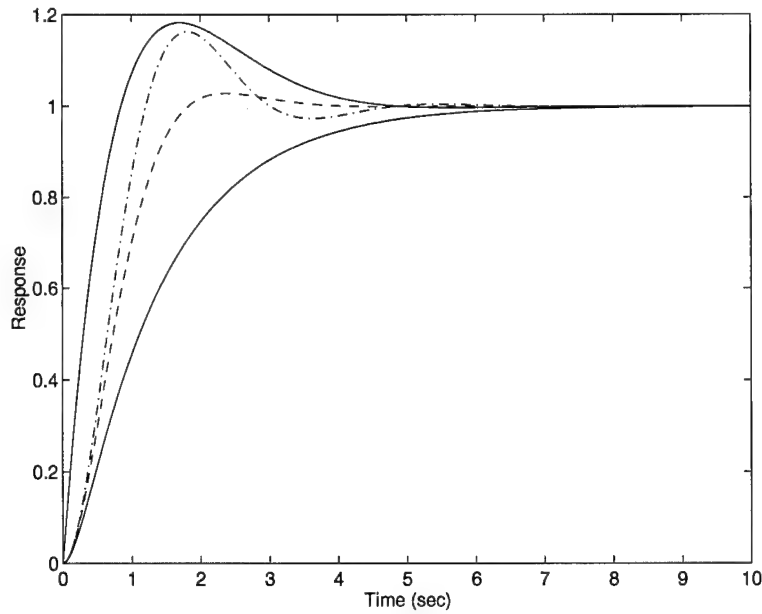
Figure 3.2   Acceptable Thumb Print Specification and Example Responses

is limited in use for optimization. While output error is the most common measure used in optimization, it is inappropriate in transient analysis since it has no inherent means for dealing with acceptable variations in the response. Recall that optimization entails the minimization of a scalar variable. That is, an otherwise very good response may be rejected because it differs too much from the reference response. Output error is particularly sensitive to time lags or a bias between two responses. For example, given a response and a duplicate of itself shifted slightly in either the x (time) or y (magnitude) directions. While these responses are nearly identical, the output error can be rather large. An attempt to avoid the problems of output error is the use of an acceptable thumb print specifications such as that in Figure 3.2, where a response is judged acceptable if it lies within the envelope formed by the two boundary responses (solid curves). However, this technique does not detect 'bad' responses within the envelope, such as highly oscillatory ones.

To avoid such shortcomings, and provide a transient analysis constraint useful in the optimization, a novel constraint functional is developed which attains the merits of the above mentioned techniques. First, a vector valued mapping, F, from the system response $y(t)$ to its feature space is constructed. This is compared to a

3-9

specification vector, η, and the results are weighted and summed. The specification vector is comprised of classical figures of merit as well as other meaningful measures and its dimension is decided upon by the designer. The weights give a degree of freedom to highlight importance of individual specifications and provide directionality to search algorithms. These weights may be constants or nonlinear functions themselves. In the examples contained in the sequel, eight tracking specifications are checked and hence the feature space is $\mathbb{R}^8$. Eight functionals are derived to extract the tracking features of each response, each an element of the feature vector, ν.

$$F_i(y(t)) = \nu_i \qquad (3.10)$$

Then a threshold function, T, is applied to the feature vector ν comparing it to the specifications η i.e., $T(\nu) = \gamma \in \mathbb{R}^8$. Thus,

$$\gamma = T(\nu) = \begin{cases} |\nu - \eta| & , \quad \nu \geq \eta \\ 0 & , \quad \nu < \eta \end{cases} \qquad (3.11)$$

Finally, the weighing functions are evaluated to $w \in \mathbb{R}^8$ and the inner product with γ gives the constraint as $C(y(t)) = w^T \gamma$, or

$$C(y(t)) = w^T T(F(y(t))) \qquad (3.12)$$

The values used for w and η, to evaluate tracking performance, as well as the definitions of η, are given in Table 3.1.

A subtle point, which often goes unaddressed in gauging MISO system performance, is now presented. Evaluating the output response in multiple input systems is not as straight forward as in the SISO case. With a single output and single input, one desires to either track a commanded input or have the system not react to some external input (disturbance rejection). Up until this point, the emphasis has been

i	w	η	Specification
1	1	0.98	-1 × Minimum % peak value
2	1	1.25	Maximum % peak value
3	5	4	Maximum number of extrema prior to settle time
4	1	0.4	Maximum ratio of consecutive extrema
5	1	1.0	Maximum % tracking error
6	1	10	Maximum time of peak value
7	1	5	Maximum 2% settle time
8	1	10	Maximum rise time

Table 3.1   Tracking Specifications and Weights Used

tracking. In multiple input systems, this is handled by applying decoupled inputs, usually one command at a time, and comparing each response to different specifications depending on the input. This is somewhat unrealistic in real world operation but allows for a level of performance evaluation. This technique can be used when all inputs are to be tracked, but requires uncertain parameters to be fixed during the simulation. It is also appropriate when all commands are zero and response to only parameter variation is investigated in a disturbance rejection problem. In the case of MISO scheduling which includes both a commanded tracking input and varying parameters, this means of response evaluation is indeed inappropriate. In such a case the specification should indicate a blending of tracking and disturbance rejection specifications.

The objective of this research is to traverse the parameter space which includes both physical variables to be tracked, and structured uncertain parameters. Therefore, such a blending of specifications is required. If the parameters were restricted to being fixed, one would actually be using a discrete parameter as the truth model and may as well reduce the order of the parameter space. The approach taken in this technique to handle the blending is to use piece-wise linear functions to yield a continuous mapping from tracking to disturbance rejection specifications, based upon the slope of the system's trajectory in the parameter space $\mathbf{P}$. Care must be taken in generating this mapping since many tracking specifications become unde-

fined in the disturbance rejection (zero commanded input) setting. Modifying the weights would be an inappropriate solution, since a constrained optimization is used and due to the form of the constraint functional, this would merely change the slope information for the optimization routine.

The MATLAB® functions used to extract the features (`fom_nl.m`) and calculate the constraints (`fuz_cost.m`) are contained in Appendix C. This two stage evaluation of the constraints allows for modification of the specifications dependent upon command strength and the amount of parameter variation during the simulation.

The constraint functional now identifies responses which are clearly acceptable, yet it does not catch all responses that should be judged satisfactory, by perhaps visual inspection, due to the finite dimension of the feature space. Hence, the solution is more conservative than it could be and somewhat suboptimal. The multi-dimensional scheduler example to follow uses the developed constraint functional of this research. However, the constraint functional does yield a multitude of responses which could be used as references and an output error measure could be meaningfully applied to these. This is examined in an example in Chapter IV.

3.4   Selection of Membership Function Variance Parameters

The final portion of the synthesis technique is the selection of membership function parameters to describe which region of the parameter space is best suited to represent each point controller. Having chosen the multivariate Gaussian MFs of Chapter II centered at the point p_i, at which a controller is designed, the real, symmetric and positive definite variance parameter matrix $\mathbf{R}_i$ must be chosen. This is accomplished in a manner which approximates the Voronoi diagram of the set of points G. The resulting scheduling surface can be viewed as a Fuzzy Voronoi diagram. Throughout this development it is assumed that all points in G are unique.

The novel method of selecting these parameters, developed in this research, addresses several shortcomings of current techniques. First, a closed form solution can be obtained, avoiding trial and error. Second, the physical meaning of membership function is maintained. Often in finding MFs by optimization, this is lost by the numeric algorithm even though this physical meaning of fuzzy variables is a main reason for using Fuzzy Logic in the first place. Finally, the effects of normalizing the weights over the entire universe of discourse is addressed. These issues can be best visualized by first digressing to a 1-dimensional problem where each membership function has a scalar variance parameter σ_i^2 to be chosen. Define $r \in \mathbb{R}^N$ as the vector whose elements are the variance parameters for each of the membership functions.

The selection of the elements of r could be addressed in one of two ways. It may be treated as another parameter in the overall optimization, or generated automatically by some rule. Unconstrained optimization on r can take away from the physical meaning of membership function and therefore is not used. Generation of a meaningful constraint yields an elegant solution. The selected procedure is based upon the physical meaning of the parameter and the effects it has on the normalized weights of the controllers. The membership function, μ_i, is used to specify the degree of membership an operating point has with respect to the fuzzy variable $\mathcal{P}_i$. Where $\mathcal{P}_i$ represents the ith point controller. This membership must be unity at p_i and near zero at the other elements of G, since there are controllers for these operating points which are designed for that point, and should therefore be used instead. Define the *cross-membership* of p_j with respect to p_i as $\mu_i(p_j)$ where $i \neq j$. Thus, one should choose the elements of the vector r such that all cross-memberships are below some threshold $\epsilon > 0$. Since normalized weights are used, this scheme converges to hard switching between controllers as $\epsilon \to 0$. Since the primary objective is smooth switching, a lower bound should also be placed on ϵ and

hence r_i. This is accomplished by selecting $r_i = \sigma_i^2$ such that

$$\max_{p_j \in G} \ \mu_i(p_j) = \epsilon, \quad j \neq i \tag{3.13}$$

Now turn to the selection of a value of ϵ. If the cross-membership (cross coverage) is too large, the normalized weights are inappropriately low at the center points as in Figure 3.3a. A value of $\epsilon = 0.001$ is chosen, from empirical analysis, and representative results are shown in Figure 3.3b. Figure 3.3c shows what can happen to the normalized weights when the effect of cross-membership is ignored.

This selection has the desired attributes by approximating the Voronoi diagram of points in $\mathbb{R}^1$. Clearly in the 1-dimensional case, the only possible active constraints are the point's nearest neighbors, those to the left and/or right of p_i on the real line. Since the Gaussian function is symmetric, the active constraint is its closest neighbor $p_c \in G$ and from there one needs to solve for σ_i^2 as

$$\sigma_i^2 = -.5 \frac{(p_i - p_c)^2}{\ln \epsilon} \tag{3.14}$$

The extension to the generalized n-dimensional case is obtained by realizing that the above procedure maximizes the length (volume) of the constant membership interval (contour) of $\mu(x) = \epsilon$ where $x \in X \subset P$. The set X is the convex region containing p_i such that no other point design is contained within its interior. Also, the only active constraints were Delaunay neighbors of p_i.

For the general case, the solution is to maximize the volume of the n-dimensional ellipsoid formed by the membership function contour

$$\mu(x) = \epsilon, \quad x \in X \tag{3.15}$$
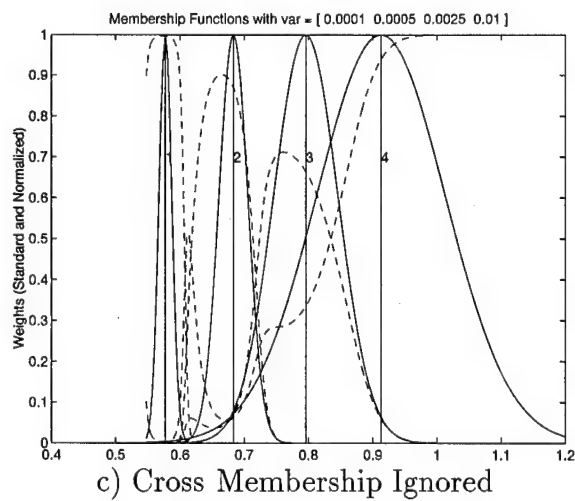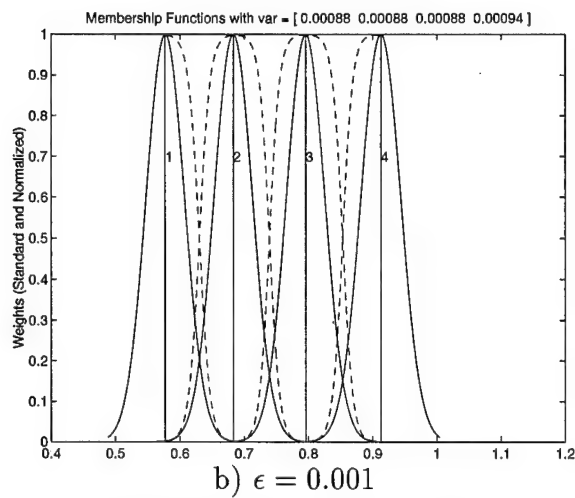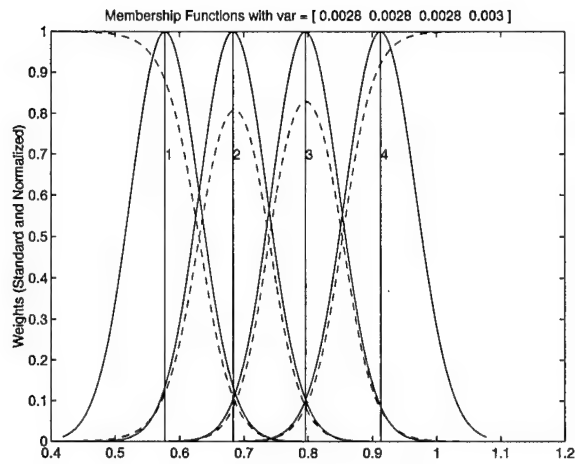
Figure 3.3   Implications of Selection of ϵ on Normalized Weights

where $X \subset P$ is convex and

$$p_i \in X \tag{3.16}$$

$$p_j \notin \text{int } X, \quad p_j \in G, \quad j \neq i \tag{3.17}$$

It is desired that the active constraints be the neighbors from the Delaunay diagram, however this is not always feasible. In the development which follows, two additional constraints are placed on the optimization.

1. The points possessing the minimum Euclidean distance from p_i are constrained to lie on the constant membership contour. Relaxation of this requirement can yield undesirable solutions.

2. The variance parameter matrix $\mathbf{R}_i$ is diagonal. This assumption makes for a much cleaner solution while its implications on the shape of the contour are reduced by the normalization of the membership functions over P.

Note that in dimensions greater than one, the above optimization may still be ill conditioned due to the location of the elements of G. In such cases the optimum solution may give infinite or degenerate (trivial) results, and additional constraints must be added for meaningful results. This is in part due to the form of $\mathbf{R}$ chosen, but is easily remedied. Possible choices are further restricting X to the convex hull of G or to the region bounded by the extreme projections of all points onto $\mathbb{R}^n$.

3.4.1 2-Dimensional Variance Solution. To illustrate the selection of the variance parameters, the solution for a set of points in $\mathbb{R}^2$, to be used in the following example, is derived. The resultant constraint membership contours are now ellipses aligned with the x and y axes, of which the constrained maximal area is desired, resulting in the following non-convex optimization.

The solution is found by translating the point of interest to the origin and removing it from consideration. Denote the remaining points as S where

$$S = \{(x_j, y_j)\}_{j=1}^{N-1}$$
$$(0,0) \notin S$$

and define

$$(\bar{x}, \bar{y}) \equiv \arg \min_{S} \|(x_j, y_j)\| \tag{3.18}$$

Then to maximize the area of the ellipse with semi-axes a, b

$$\max_{a,b \in \mathbb{R}^+} ab, \tag{3.19}$$

such that

$$\frac{x_j^2}{a^2} + \frac{y_j^2}{b^2} \geq 1 \quad \forall j \tag{3.20}$$

$$\frac{\bar{x}^2}{a^2} + \frac{\bar{y}^2}{b^2} = 1 \tag{3.21}$$

To assure finite non-degenerate solutions, restrict $a_{\min} \leq a \leq a_{\max}$ and $b_{\min} \leq b \leq b_{\max}$ with $a_{\min}, b_{\min} > 0$. For a well conditioned set of points, these bounding values are achieved by the Delaunay neighbors and other elements of S. If not, they must be imposed, perhaps as suggested above. Now define

$$x \equiv \frac{1}{a^2} \tag{3.22}$$

$$y \equiv \frac{1}{b^2} \tag{3.23}$$

Clearly $x, y \in \mathbb{R}^+$ and $\max ab \iff \min xy$ and the limits on a, b impose limits on x, y. The new constraints are

$$x_j^2 x + y_j^2 y \geq 1 \quad \forall j \tag{3.24}$$

$$\bar{x}^2 x + \bar{y}^2 y \;=\; 1 \tag{3.25}$$

The isocost surfaces of $\min xy$ are hyperbolas, symmetric about the line $x = y$ in the positive cone of $\mathbb{R}^2$, which have local solutions at $x^*_{\min}$ and $y^*_{\min}$. These refined minima, from those imposed to generate finite solutions, are developed below. From the equality constraint,

$$y = \frac{1 - \bar{x}^2 x}{\bar{y}^2} \tag{3.26}$$

yielding the combined inequality constraint on x as

$$\left[x_j^2 - \frac{y_j^2}{\bar{y}^2}\bar{x}^2 \right] x \geq 1 - \frac{y_j^2}{\bar{y}^2} \quad \forall\, j \tag{3.27}$$

denote the above quantities such that

$$\alpha_j x \geq \beta_j \quad \forall\, j \tag{3.28}$$

It is easily verified that α may change sign requiring

$$x \;\geq\; \frac{\beta_j}{\alpha_j} \quad \text{for} \quad \alpha_j > 0 \tag{3.29}$$

$$x \;\leq\; \frac{\beta_j}{\alpha_j} \quad \text{for} \quad \alpha_j < 0 \tag{3.30}$$

A parallel development results in constraints on y. In the sequel, it is shown that $\alpha_j = 0$ need not be addressed. Combining the, at most $N-1$, evaluations of of these constraints for both x and y along with the original $x_{\min}$ $(a_{\max})$ and $y_{\min}$ $(b_{\max})$ yield $x^*_{\min}$ and $y^*_{\min}$ giving solutions

$$(x^*, y^*) \;=\; \left(x^*_{\min}, \frac{1 - \bar{x}^2 x^*_{\min}}{\bar{y}^2} \right)$$

$$\text{or}$$

$$(x^*, y^*) \;=\; \left(\frac{1 - \bar{y}^2 y^*_{\min}}{\bar{x}^2}, y^*_{\min} \right)$$

the global minimum being that which yields the smallest value of x^*y^*. This gives

$$a^* = \sqrt{\frac{1}{x^*}}$$

$$b^* = \sqrt{\frac{1}{y^*}}$$

Since $\mathbf{R}_j$ is diagonal, these values can be used twice in Eq. (3.14) where a^* and b^* represent the quantity $(p_i - p_c)$, thus totally describing the membership function with desired cross-membership. The MATLAB® function `find_2dv.m` which performs this optimization is contained in Appendix C

To visualize this process, examples of cross-membership contours and the resulting membership functions, both regular and normalized, for the set of points in $\mathbb{R}^2$ used in Figure 3.1 are contained in Figure 3.4. In particular, Figure 3.4d is a fuzzified version of Figure 3.1a.

Claim: For $\alpha_j = 0$, the point (x_j, y_j) can be removed from the constraints.

Proof: From the definition of α_j

$$\alpha_j = 0 \implies \frac{x_j^2}{y_j^2} = \frac{\bar{x}^2}{\bar{y}^2}$$

$$\implies \left|\frac{x_j}{y_j}\right| = \left|\frac{\bar{x}}{\bar{y}}\right|$$

$$\implies \frac{x_j}{y_j} = \left|\frac{\bar{x}}{\bar{y}}\right| \quad \text{or} \quad -\left|\frac{\bar{x}}{\bar{y}}\right|$$

This implies that (x_j, y_j) lies on either the line L, the line through $(0,0)$ and $(\bar{x}, \bar{y})$, or L', the reflection of L about an axis. From the definition of $(\bar{x}, \bar{y})$

$$\|(x_j, y_j)\| \geq \|(\bar{x}, \bar{y})\|$$

Thus:

a) Local Cross Membership Contours

b) Solution Contour Map

c) 2-D Membership Functions

d) Normalized Membership Functions
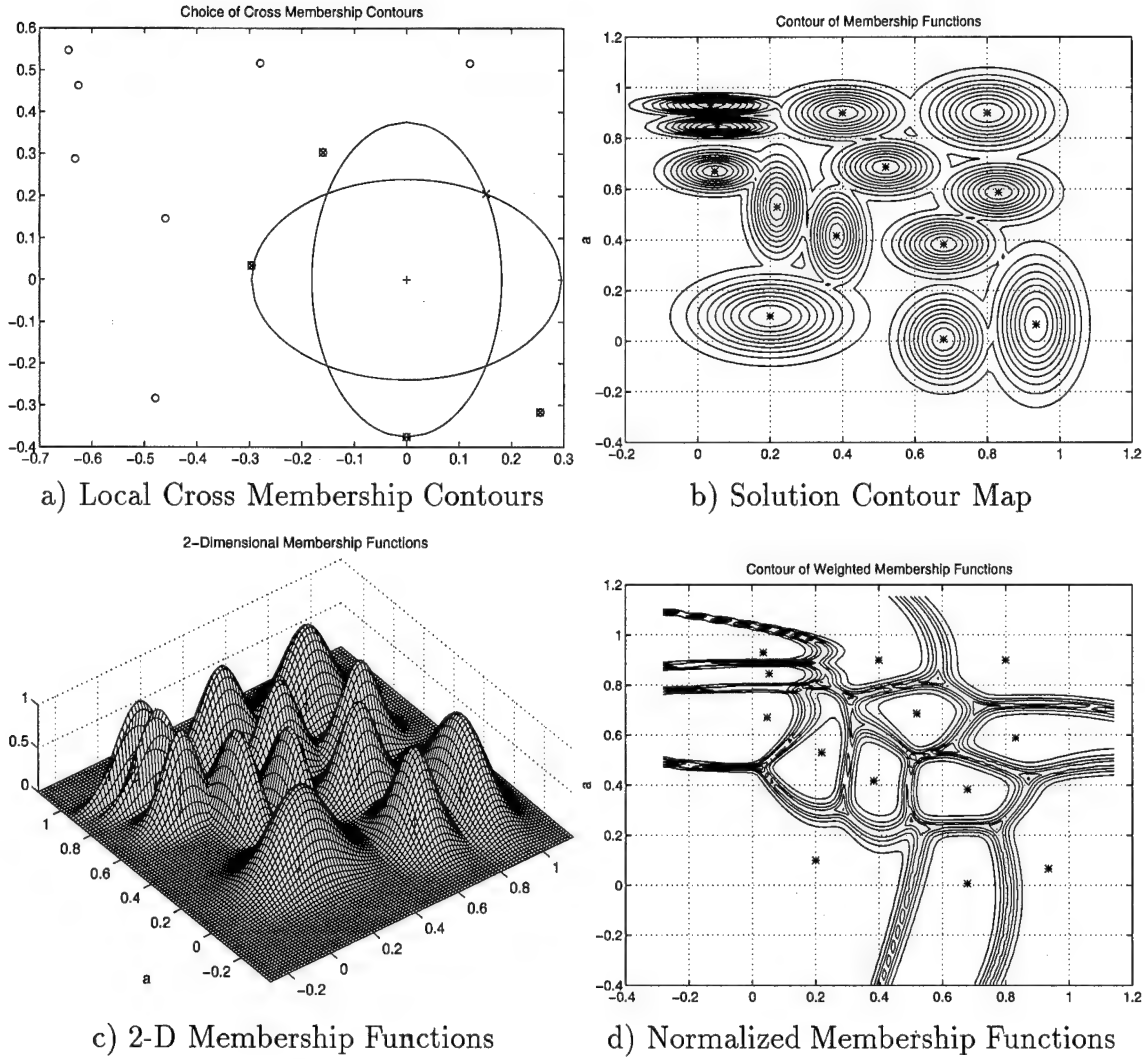
Figure 3.4   Membership Function Selection of Points in $\mathbb{R}^2$

1. If equality holds, point (x_j, y_j) lies on the circle of radius $\|\bar{x}, \bar{y}\|$. The constraint is equivalent to the one on $(\bar{x}, \bar{y})$ and can therefore be ignored.

2. If $\|(x_j, y_j)\| > \|(\bar{x}, \bar{y})\|$, the point is clearly outside the circle of radius $\|\bar{x}, \bar{y}\|$ while being on L or L'. This implies

$$|x_j| > |\bar{x}| \quad \text{and} \quad |y_j| > |\bar{y}|$$

$$\implies x_j^2 > \bar{x}^2 \quad \text{and} \quad y_j^2 > \bar{y}^2$$

Since $x, y > 0$ by definition, this gives

$$x_j^2 x + y_j^2 y > \bar{x}^2 x + \bar{y}^2 y = 1, \ \forall \ x, y > 0$$

Therefore:

Point (x_j, y_j) lies outside the family of possible ellipses and can be removed from the constraints.

■

3.5   2-Dimensional Scheduler Example

With the mathematical formulation in hand, a solution to scheduling 10 LTI point controllers to control the nonlinear system, of Appendix B, with parameter uncertainty is obtained.

3.5.1   Plant and Point Controllers.   The nonlinear system with uncertain parameter, a, of interest is described by the system of equations

$$\dot{x}_{f_1} = -x_{f_1} + ax_{f_2} \tag{3.31}$$

$$\dot{x}_{f_2} = -x_{f_1}^3 + u_f^3 \tag{3.32}$$

$$y_f = x_{f_1} \tag{3.33}$$

To linearize these equations define the full state x_f as

$$x_f = \bar{x} + x$$

where $\bar{x}$ is the trim state and x is the perturbation state. Solving for the trim condition, the model yields $\bar{x}_1 = a\bar{x}_2 = \bar{u}$. Further, define $\tau = 3\bar{x}_1^2 = 3\bar{u}^2$. This gives

the perturbation equations from the equilibrium point $\bar{x}$ as

$$\dot{x}_1 = -x_1 + ax_2 \tag{3.34}$$

$$\dot{x}_2 = -\tau x_1 + \tau u \tag{3.35}$$

$$y = x_1 \tag{3.36}$$

When the two parameters τ and a are considered fixed, an LTI model results. For a given set of these parameters, $p = (\tau, a)$, the transfer function representation of the plant, at the trim condition, as a function of τ and a is

$$P_{\tau,a}(s) = \frac{a\tau}{s^2 + s + a\tau} \tag{3.37}$$

The scheduling solution (location of point controllers) is found by a numeric optimization, and the search algorithm requires several point evaluations for each step along the cost surface to determine the direction and size of step. Each point evaluation requires the design of N LTI point controllers, their corresponding membership functions, and simulations of the resulting system to evaluate the constraint. These procedures are automated to allow for a hands off optimization, and all have been addressed with the exception of point controller design method which is left to the discretion of the designer.

The design of LTI point controllers can be by any technique not requiring an interactive approach, such as LQR or pole placement. Using the linear control theory analysis of Appendix B as guidance, the scheduled controllers $G_i(s)$ designed at point $p_i = (\tau_i, a_i)$, are of the form

$$G_i(s) = \frac{\frac{24}{a_i \tau_i}(s^2 + s + a_i \tau_i)}{s(s+4)(s+4)} \tag{3.38}$$
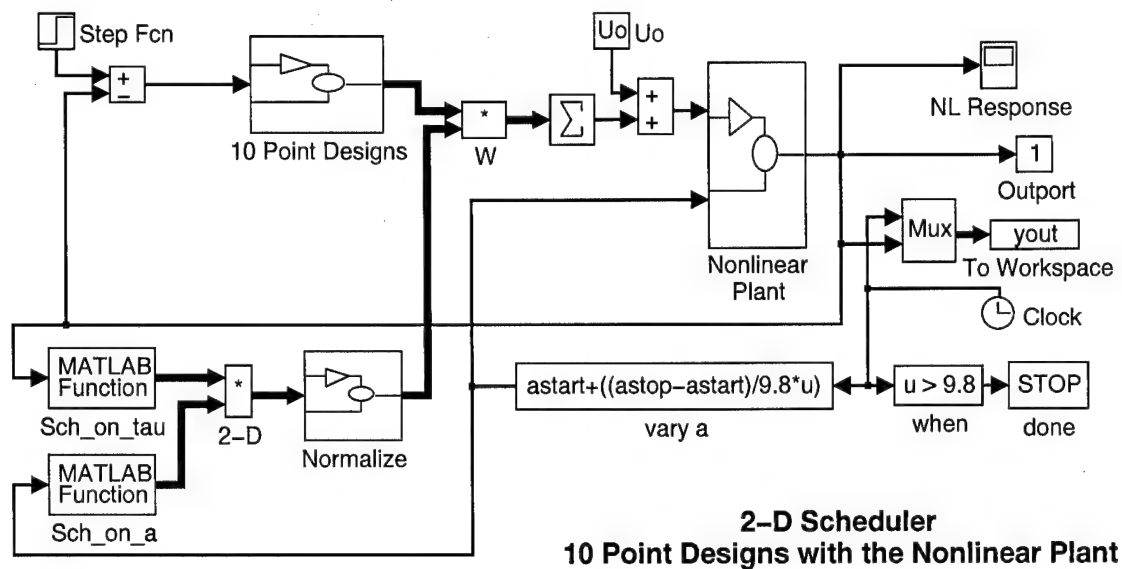
Figure 3.5   Block Diagram of Fuzzy Scheduler
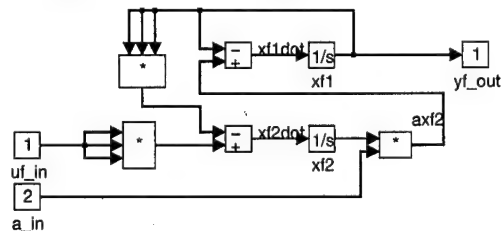
Figure 3.6   Nonlinear Plant with Parameter Uncertainty

3.5.2   The Fuzzy Scheduler.   The fuzzy scheduler to control the plant over some region of the parameter space P is depicted in Figures 3.5 - 3.7. The number of LTI point controllers is arbitrarily chosen as $N = 10$.

Some notes on the problem as modeled in Figure 3.5 are in order. Traversing P requires varying both of the parameters, τ and a, during the simulation. Recall that $\tau = 3\bar{x}_1^2$ is only used to generate approximate LTI models of the plant to aid in

3-23
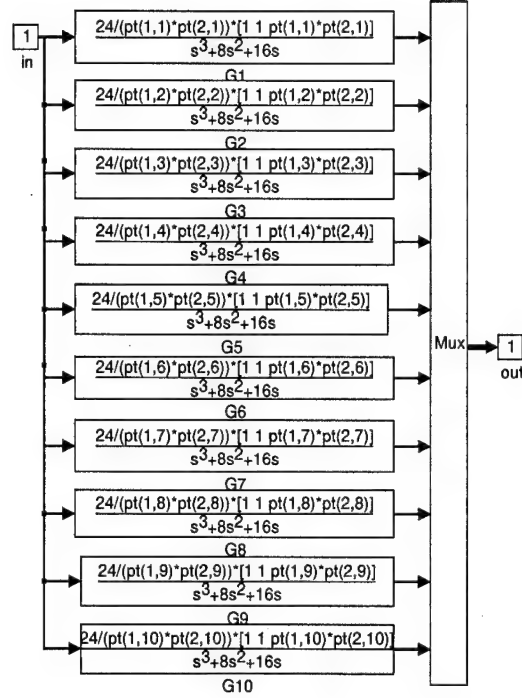
Figure 3.7   Bank of 10 Point Designs

the design of the point controllers. It is therefore advantageous to consider a varying τ as $\tau = x_1^2$ when dealing with non-LTI systems. Since $x_1 > 0$, an equivalency is established between the two representations of P as either $[\tau \times a]$ or $[x_1 \times a]$. The implementation actually schedules on x_1, the output of the nonlinear system. This is consistent, since the membership functions are actually found in the mapped space $[x_1 \times a]$. So the Sch_on_tau block in the figure is actually scheduling on the output y as desired. This notation is maintained throughout the research.

The value Uo $= \bar{x}_1$ is the required command input to achieve trim, thus avoiding the calculation of controller state initial conditions. Appropriate initial conditions are calculated for the nonlinear plant's integrators. Changes in τ are clearly accomplished with steps commands of x_1, while the required changes in a are modeled as ramping from the initial $\bar{a}$ to the final value of a_f over the duration of the simulation. Thus, a varies as is expected for a slowly changing external parameter over which the system has no control authority. This problem is specifically designed to generate such requirements and validate the synthesis technique's ability to handle them. As such, the specifications are modified from the pure tracking of Table 3.1, when $\bar{a} = a_f$, to pure disturbance rejection when the step function has zero strength.

To clarify, the two blocks Sch_on_tau and Sch_on_a do not indicate independent scheduling of elements of P as is done in the current literature noted in Section 1.5. This is merely a result of assuming $\mathbf{R}_i$ of the membership functionals to be diagonal and is implemented for speed of simulation. As presented earlier, although the cross-membership contours are aligned with the basis axes, the solution of the membership function fit is performed in the full n-dimensions.

3.5.3 Optimization. Now that the problem statement is fully posed, the optimization can be performed using any of a number of techniques. The optimization is performed by the *Optimization Toolbox's* function constr.m for MATLAB® , an SQP routine [28]. The simulations to evaluate performance constraints are obtained from SIMULINK® . The algorithm to achieve the solution is:

1. Generate the argument of the optimization $\mathbf{X} \in \mathbb{R}^{N \times n}$ composed of the N elements of G, each describing a point controller's location.

2. Calculate the object function. That is the area of the convex hull of the G.

3. Generate the N point controllers $G_i(s)$.

4. Solve for the Delaunay neighbors of each of the N point controllers.

5. Solve for the membership function variance parameter matrices. Schedule on $[x_1 \times a]$.

6. Evaluate the constraints. For each point in G:

   (a) Trim the model to $(\bar{x}_i, \bar{a}_i)$

   (b) Calculate the step strength, change in a, and slope of the trajectory in P from point p_i to each of its Delaunay neighbors.

   (c) Simulate the scheduled system from point p_i to each neighbor.

   (d) And for each response: Extract the response features, then calculate appropriate specifications, and evaluate the constraints of slewing to that neighbor. If no violations are found, the slewing goal is achieved between all neighbors and sufficient cover is demonstrated.

7. Return the values of object function and constraint evaluations.

8. Update the argument with a new set of point controller centers.

3.5.4 Solution. The results of the optimization for $N = 10$ are presented, demonstrating the feasibility of the fuzzy scheduler. The objective of the optimization is to maximize the cover of the controller, while meeting performance constraints when slewing between controllers. Figure 3.8 shows an increase in area over 50 times larger than the initial location of controllers as supported by the two shaded areas.

All performance specifications are met (see Table 3.1 page 3-11) as indicated by the optimization's constraint functional evaluating to $C(y(t)) = 0$ at the solution. The slewing responses of commanded input between all point designs are shown in Figure 3.9

The issue of system stability is addressed by extensive simulation in the region of P defined by the convex hull of the solution G^*. The spanning set of points, S, in Figure 3.10a is used for this purpose. A simulation is started from each $s_i \in S$ to all
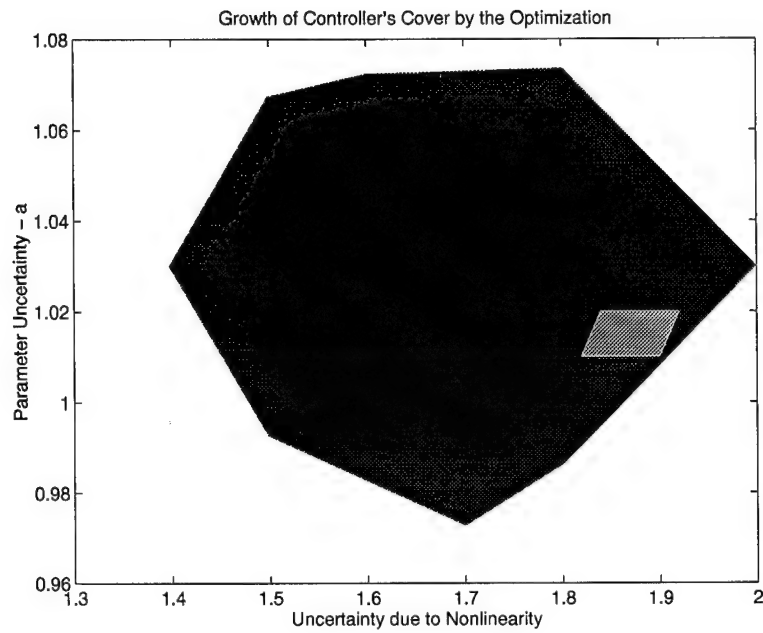
Figure 3.8   Increase in Object Function
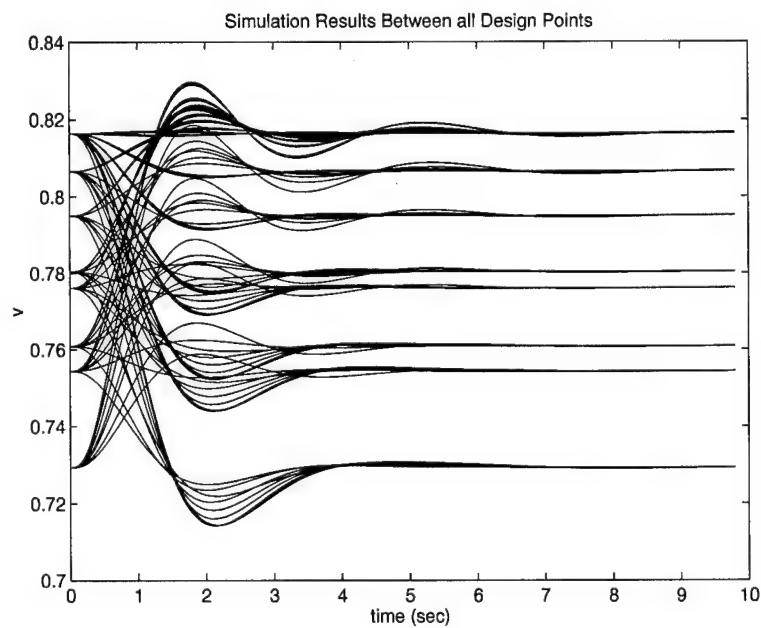
Figure 3.9   Tracking Response of Slewing Commands

3-27

a) Spanning Set of Points S   b) Tracking Responses
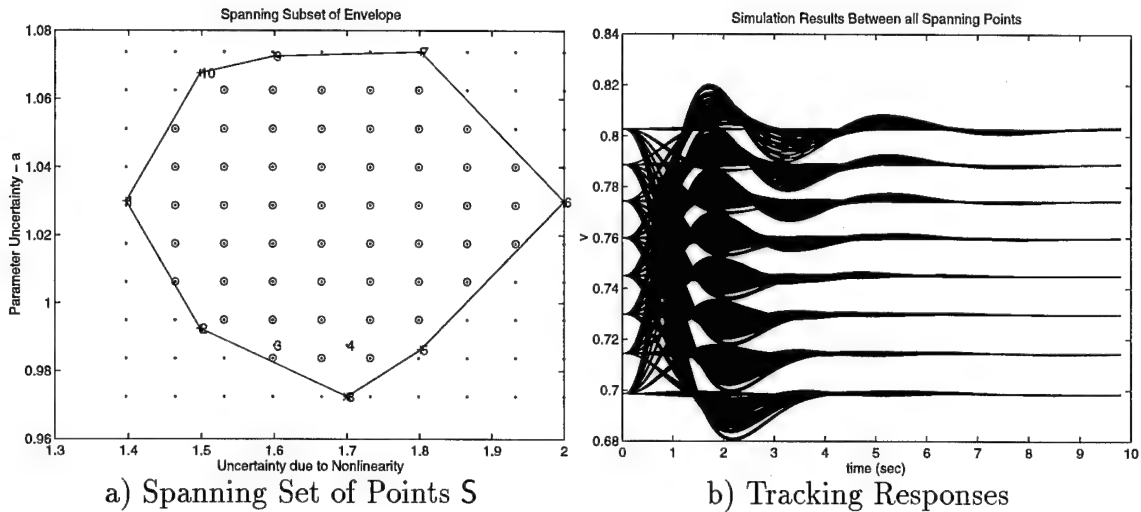
Figure 3.10   Stability Analysis

other admissible points. Since the region of interest is convex, the admissible set in-
cludes all other points of S. The results of these simulations are in Figure 3.10b. The
tracking specifications are not applied to these responses since only bounded steady-
state error is required for stability. However, the results are more than satisfactory
and clearly indicate a smooth transition between the controllers as desired.

Although this example gives total freedom to the optimization for the location
of the point controllers, the synthesis technique has the ability to direct these loca-
tions. One merely needs to include additional fixed points to the bank of controllers
to achieve this. These points are not included in the argument of the optimization
routine, but included in the membership function generation, simulation and con-
straint evaluation portion of the algorithm. This allows the designer the freedom to
include specific points of concern in P as well as firmly establishing the boundaries
of the operational envelope. This can also allow the design to grow by fixing the
results of one optimization, then adding new free controllers as the new argument of
the optimization. An example of directing the solution is contained in Chapter IV.

To gain greater insight into the power of this technique, Chapter IV includes 1-dimensional examples with extensive analysis of the results. These examples include varying the fidelity of the truth model and its implications.

3.6   Summary

This Chapter develops the fuzzy scheduler of point controllers to provide full envelope control of nonlinear systems with parameter uncertainty. Scheduling is performed on the n-dimensional parameter space, P, over which the system is to operate. The scheduling is performed to provide a smooth transition between point controllers designed for specific regions of P. The solution is obtained from an optimization which yields the point controller locations, and membership functions which maximize the coverage of the scheduled controller while providing a guaranteed level of performance. The sufficient cover required of the point designs is quantified as a by product of the optimization process. The resulting controller is for use with MISO systems. The solution can therefore be used in conjunction with MIMO controller synthesis techniques which yield solutions by solving several MISO problems, thus bridging the gap to full envelope control of MIMO nonlinear systems with parametric variation.

The developed synthesis technique allows one to use existing linear control techniques to design point controllers yielding satisfactory results for non-LTI plants. The use of standard linear control theory for the point controllers removes the restriction of controlling only minimum phase and/or simple plants inherent in the standard application of fuzzy logic based on control. This ability to handle such plants is designed into the point controllers and is not violated by an application of error and error rate in the fuzzy logic inference engine. The resultant controller is able to perform high amplitude (not small signal) control, and the nonlinear plant can be slewed.

To obtain the solution, several novel concepts are introduced to the field. The use of the Voronoi diagram to quantify generalized nearest neighbors fits naturally into n-dimensional scheduling. The concept of cross-membership ellipsoids maintains the physical meaning of fuzzy variables while allowing an optimization to calculate the membership function parameters. When the MFs are obtained in this manner, normalizing the results provides a means of approximating the Voronoi diagram as an analytic scheduling/switching surface over P. This also assures sufficient control authority as the scheduling dimension increases.

The direct attack on both model uncertainty due to nonlinearities and uncertainty of varying parameters points out the inadequacies of pure tracking/disturbance rejection specifications. The ability to achieve the demonstrated results is based upon formulating a constraint functional on the output response which embeds both classical figures of merit and other meaningful features. This functional has the ability to smoothly blend the tracking and disturbance specifications required to traverse the parameter space.

Finally, the applicability of the synthesis technique is demonstrated by the solution of a nontrivial strongly nonlinear control problem. This entails the solution of a 2-dimensional scheduling task of a strongly nonlinear plant with a varying parameter. The results provide an improvement in the coverage of the scheduled system over that obtainable from the point designs used in a switching scheme. All this is achieved while maintaining the pre-specified level of performance.

IV. 1-Dimensional Scheduling Examples

In Chapter III the solution to the general n-dimensional scheduling problem is developed and a non-trivial two-dimensional example is presented. To provide greater insight into the results of the fuzzy controller scheduler, simplifications are made to the plant and new solutions are obtained. The one-dimensional problems posed in Appendix B are now provided in full detail. Also included is a design of a lateral coordinated turn controller for an Air Force C-135 transport aircraft with restricted model information. This lack of information ties the hands of the scheduler and dose not allow for full freedom in the placement of point controllers. However, the ability to smoothly schedule conventional controllers used in flight control demonstrates the wide applicability of the technique.

In Appendix B experiments are performed to evaluate fuzzy scheduling of independent point controllers to obtain a full envelope controller. Results are obtained for Linear Time-Invariant, Linear Time-Varying, and nonlinear models of a strongly nonlinear plant. In this chapter, an optimal solution for the controller, which entails scheduling on a single variable, is derived. The resulting controller addresses the shortcomings noted in Appendix B as well as quantifying the "sufficient cover" required of the point designs, and yields smooth transitions between these point designs. The n-dimensional solution is applied in particular to the one-dimensional (no parameter variation) true nonlinear system, then the technique is applied to the LTI and LTV approximations of the system for completeness.

The architecture of the controller is the same as in Chapter III except that the dependence on the varying parameter a is removed, i.e $a = 1$. The resulting block diagram for the nonlinear system is shown in Figures 4.1 - 4.3.
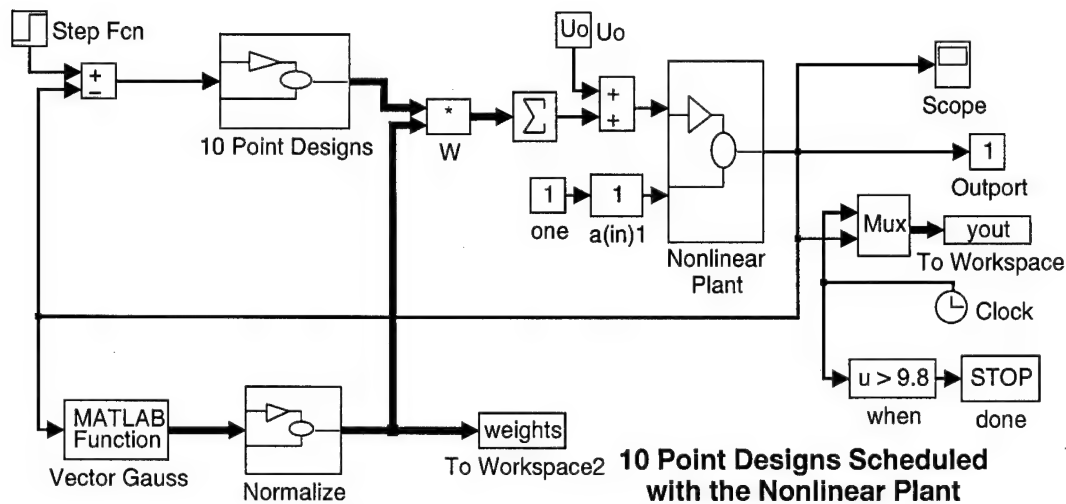
Figure 4.1   SIMULINK® Block Diagram of 10 Point Scheduler with Normalized Weights
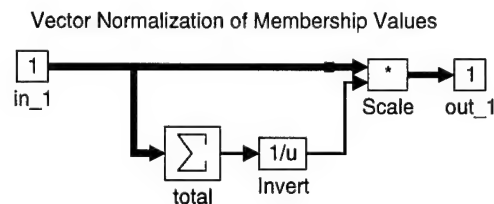
Figure 4.2   Normalization of the Weights

4.1   Simplifications Due to 1-Dimensional Scheduling

Reducing the scheduling dimension from n to one simplifies many of the scheduler's constructs. The simplifications below not only reduce the complexity of the optimization, but add a sense of directionality to the solution. Thus, one can visually appreciate the novelty of this research's contribution.
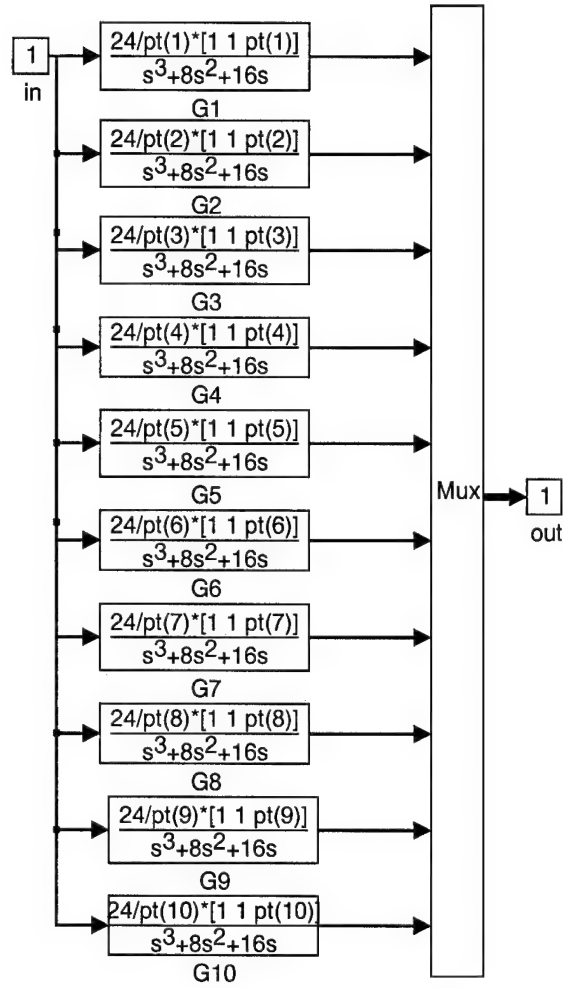
Figure 4.3   Bank of 10 Point Controllers

1. The set of point design centers $\{p_i\}_{i=1}^{N} = G$ are now represented as an element of $\mathbb{R}^N$.

2. Delaunay neighbors of p_i are merely the points to the left and/or right on $\mathbb{R}^1$.

3. Membership function variance parameters are obtained directly from Eq. (3.14).

4. The object function, or cover to be maximized, is now the length of the interval $|p_{max} - p_{min}|$.

5. Line intervals are convex, therefore all steps in its range are admissible.

6. Constraints: all commands are strictly tracking or disturbance rejection so no blending of specifications is needed. In this example all tasks are tracking.

7. The net result of the optimization is to maximize the inter-point spacing. By adhering to the cross-membership constraint this maximization is accomplished independently for the end points defining the universe of discourse. The net effect is that the 1-dimensional scheduler can be solved sequentially.

4.2   Variation on the Constraint Functional

As alluded to earlier, the constraint functional developed in Chapter III identifies responses which are clearly acceptable. It does not catch all responses that can be judged satisfactory, due to the finite dimension of the feature space. However, the functional does yield a multitude of acceptable responses that are acceptable as reference responses to which an output error measure can be applied. A modified version of the functional, which takes advantage of this property, is used in obtaining the 1-dimensional solution. As is shown in the sequel, C is not convex with regions that violate the constraint surrounded by regions that do not violate. Since it is assumed that the plant is continuous, the system's response varies continuously with respect to varying step inputs. That is for $\Phi(s|x) = y(t)$ and $\epsilon > 0$ there exist a $\delta > 0$ such that $|y(t) - \bar{y}(t)| < \epsilon$ for all $|s - \bar{s}| < \delta$ where $\Phi(\bar{s}|x) = \bar{y}(t)$. So allowing for some small output error from a known good response, allows some responses that violate the mapping of C to be accepted, and thus reduce the unmodeled features in C. This second stage of constraint evaluation is used to check unacceptable responses as evaluated by C to the 'closest response' for which $C(y(t)) = 0$. The measure used is the mean squared output error of the responses normalized to their final value.

4.3   Optimization

With the mathematical formulation in hand, the solution to scheduling LTI point controllers to control the nonlinear system of Appendix B is obtained.

4.3.1 Set Up. The parametric linear controllers of the form $G_i(s)$ of Appendix B are used. Gaussian membership functions, centered at p_i, are used and their variance parameters are found by using a cross-membership of $\epsilon = 0.001$.

To evaluate the constraint, simulations are performed starting with initial conditions at each point design, $p_i \in \mathsf{G}$. The responses are checked for maximum positive and negative steps to the closest point designs (left/right neighbors) to maximize the separation between them. The constraint, C, is not symmetric about the trim condition in general, so the limiting direction is chosen. As is shown in the simulations, this is always the positive step in the case of the nonlinear system. Once the spacing between point designs is maximized, based solely on the evaluation of C acting on the step responses from each p_i to its neighboring point designs, G is considered a candidate design for further checking. Since the constraint C is not convex, specifically on the region about p_i bounded by its neighbors, an exhaustive search is performed in this region to check for violations of C. If no violations are found, the tracking performance is achieved on this region and sufficient cover is demonstrated. If violations of C are found, a normalized output error measure is applied to these responses. The reference used is the closest normalized response which satisfies the constraint C. If the measure is less then $\epsilon = 5 \times 10^{-4}$, found empirically, the response is judged acceptable and the region has been optimized as stated. In the case where there still exists violations of the constraints after the output error check, the region is rejected and the candidate G fails due to transient performance. A note is in order here with regards to the exhaustive search on the region, since the search can not be actually performed. A finite search is used with a fine partitioning of the region, this approximation is justified by the continuity of change of the output response. This is really a numeric optimization problem and does not distract from the contribution of the overall solution.

Prior to obtaining the solution, one additional constraint is placed on the problem based upon engineering judgement. That is, $p_1 = 1$ is required since the

a) Controller Weights   b) Slewing Commands
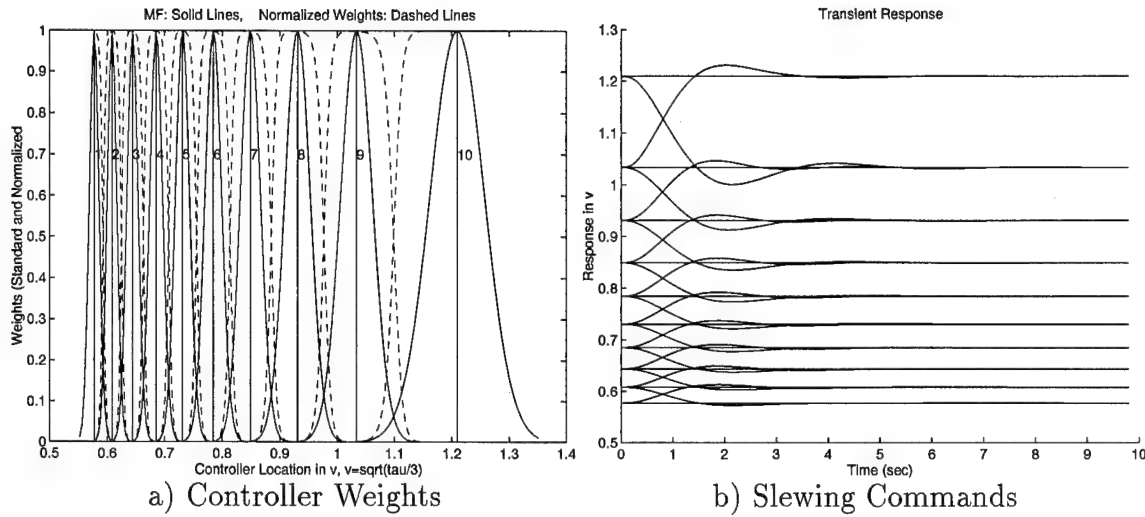
Figure 4.4   Results of Optimization

nonlinearity is most severe as the origin is approached. This way, full control authority is available at the edge of the operational envelope which causes the most trouble. This is an example of directing the synthesis technique's solution.

4.3.2 Solution.   The optimization is performed using $N = 10$ point designs which is more than sufficient to cover the interval $[1,2]$ attempted in Appendix B. The solution yields

$$p^{*^T} = [1 \quad 1.11 \quad 1.244 \quad 1.408 \quad 1.602 \quad 1.846 \quad 2.163 \quad 2.6 \quad 3.207 \quad 4.388] \quad (4.1)$$

$$\sigma^{2^{*^T}} = [6.9e{-}5 \ \ 6.9e{-}5 \ \ 9.2e{-}5 \ \ 1.2e{-}4 \ \ 1.5e{-}4 \ \ 2.1e{-}4 \ \ 3.0e{-}4 \ \ 4.8e{-}4 \ \ 7.6e{-}4 \ \ 2.2e{-}3]$$

$$(4.2)$$

where Figure 4.4a shows the controller membership functions and normalized weights. This suggests that the solution to the dual problem requires a maximum of 7 controllers of this type to cover the interval $\tau = [1,2]$. Figure 4.4b demonstrates the slewing between point controllers.

The increase in spacing of the point designs, and corresponding width of membership functions, as one moves away from the envelope bound at $p_1 = 1$ agrees

with the increase of the strength of the nonlinearity as one approaches the origin. As the nonlinearity becomes more severe, the region about each LTI model where it represents the nonlinear system is reduced. The strength of the nonlinearity can be quantified as a normalized error of the linearized model. This is depicted in Figure 4.5 for $y = x^3$ and defined as S below.

$$S \triangleq \frac{y(x + \Delta x) - y(x) - y'(x)\Delta x}{y'(x)\Delta x} \tag{4.3}$$

and Δx is fixed. Now,

$$y(x + \Delta x) = y(x) + y'(x)\Delta x + y''(x)\frac{\Delta x^2}{2} + \cdots$$

$$\implies S \propto \frac{y''(x)\Delta x^2}{2y'(x)\Delta x} = \frac{1}{2}\Delta x \frac{y''(x)}{y'(x)}$$

$$\implies S \propto \frac{1}{2}\frac{y''(x)}{y'(x)}$$

So the case of the cubic nonlinearity $y = x^3$ results in

$$S \propto \frac{1}{2}\frac{6x}{3x^2} = \frac{1}{x}$$

which is indicative of a very strong nonlinearity near the origin.

4.4 Simulation Results

The inherent directionality of the 1-dimensional problem suggests that stability be checked by commands to the extremes of the universe of discourse. The stability of the controller over $\tau \in [1, 4.388] = \mathsf{P}$ is checked by commanding the maximum positive/negative admissible steps from randomly generated trim conditions $u_0 \in \mathsf{P}$. The results in Figure 4.6 indicate a stable system over all of P.

In Appendix B an effective increase in cover is obtained by the scheduler over the individual point designs. That is, each individual controller has a region
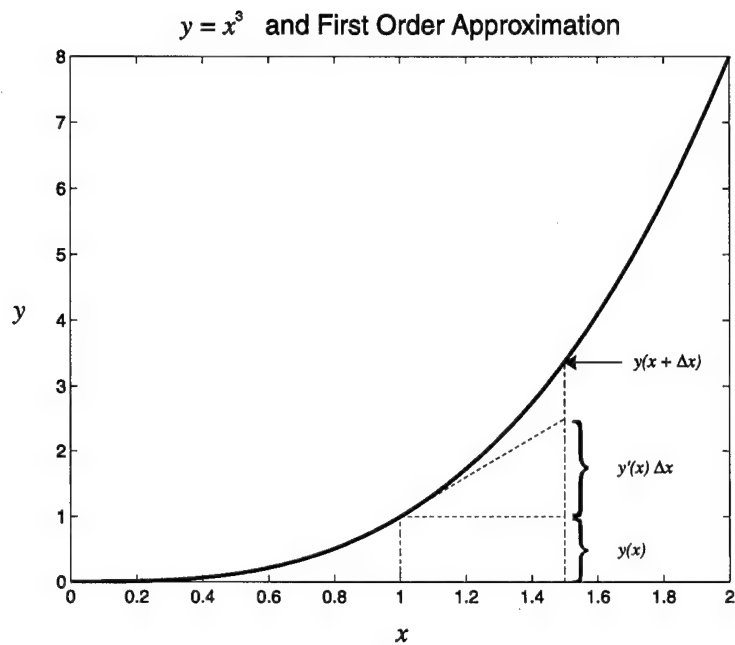
Figure 4.5   Gauging the Strength of a Nonlinearity
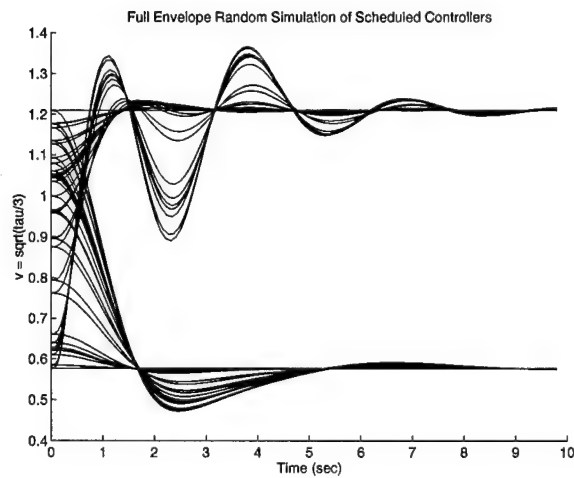
Figure 4.6   Stability Analysis of the Scheduler

$B_i(p_i, \epsilon_i) \subset \mathsf{P}$ about it for which acceptable tracking performance is obtained. With the classical view of scheduling, one would require

$$\bigcup_{i=1}^{N} B_i(p_i, \epsilon_i) \supset \mathsf{P} \tag{4.4}$$

That is, place the point designs close enough together to provide overlap of these regions. However, since this research schedules controllers and not parameters, the controller is able to often bridge the gap between disconnected regions of adjacent controllers. The set P is no longer required to be a subset of the union of the N regions. Thus allowing an increase in their separation and requiring fewer point controllers to do the job. This is observed in the n-dimensional controller of Chapter III but can be visualized much more clearly in the 1-dimensional examples. This is seen in Figure 4.7 where the error bars indicate regions of acceptable transient response for different trim conditions. Figure 4.7a shows the regions for the individual point designs which do not all have overlap. Figure 4.7b compares this with the regions of the scheduler which shows not only a drastic improvement in the regions, but also demonstrates sufficient cover and the ability to smoothly transverse point controllers. Responses corresponding to these increases in cover are contained in Appendix D.

Evaluations of the constraint surface over the entire operational envelope gives an indication of the increase in performance. Figure 4.8 gives the surfaces before and after the fuzzy scheduling.

To visualize the effects of the two step constraint evaluation, an example constraint surface for $p_{10} = 4.388$ is shown in Figure 4.9. The x-axis is $v = \sqrt{\tau/3} = x_1$, the y-axis is $C(y(t))$, the $\cdots$ lines are the location of other point designs, and the $-\cdot-$ lines indicate the region of acceptable transient response as determined by C. The symbol $*$ is used to show the value of the constraint evaluation *after* the normalized output error comparison. Constraint surfaces for other trim conditions are

a) Cover of Point Controllers   b) Cover of Scheduler
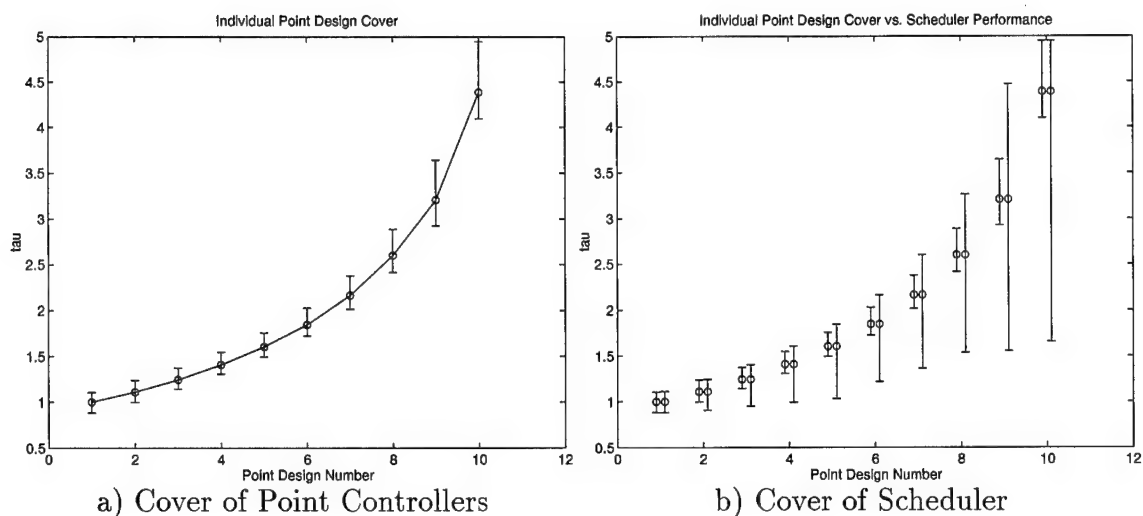
Figure 4.7   Increase in Cover

a) Point Controllers   b) Scheduler
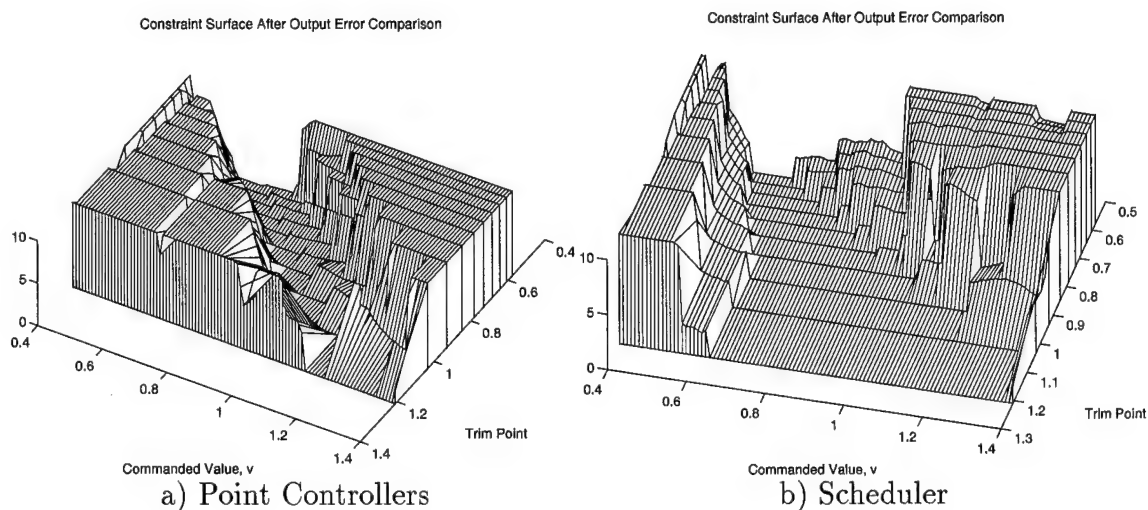
Figure 4.8   Performance Surface Before and After Scheduling

found in Appendix D. Figure 4.10 depicts an evaluation of the normalized output error measure for a response where $C(y(t)) > 0$ in the interval $[p_9, p_{10}]$.
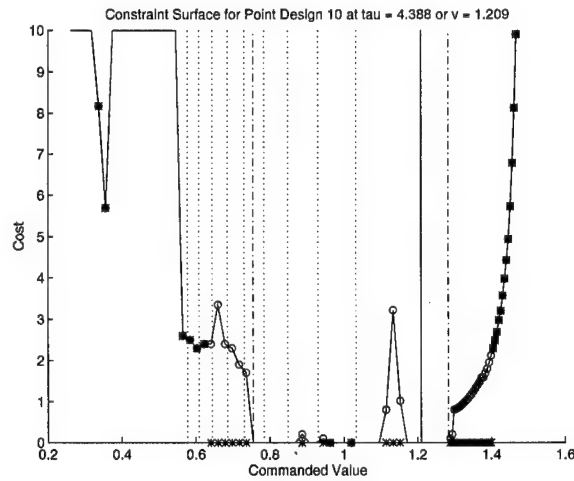
Figure 4.9   Constraint Surface for p_{10} where $u_0 = 1.209$
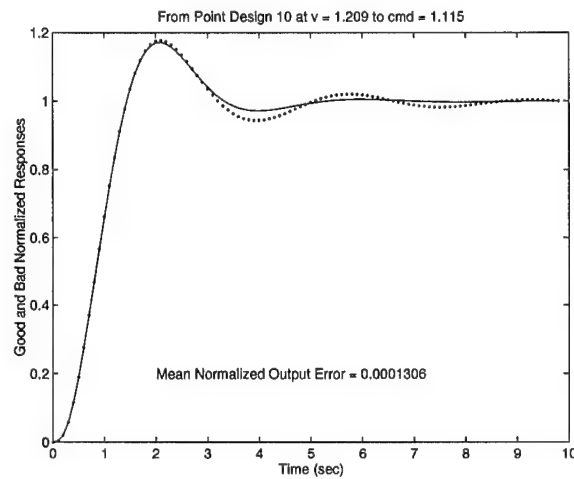
Figure 4.10   Visualization of Mean Normalized Output Error

4.5   LTI and LTV Results

The optimizations for the two systems of Appendix B, used to approximate the true nonlinear system, are performed using $N = 4$ point designs. This number allows for covering of the specified operational envelope as well as confirming the ability to slew and transverse several controllers.

4-11

a) Controller Weights   b) Slewing Commands
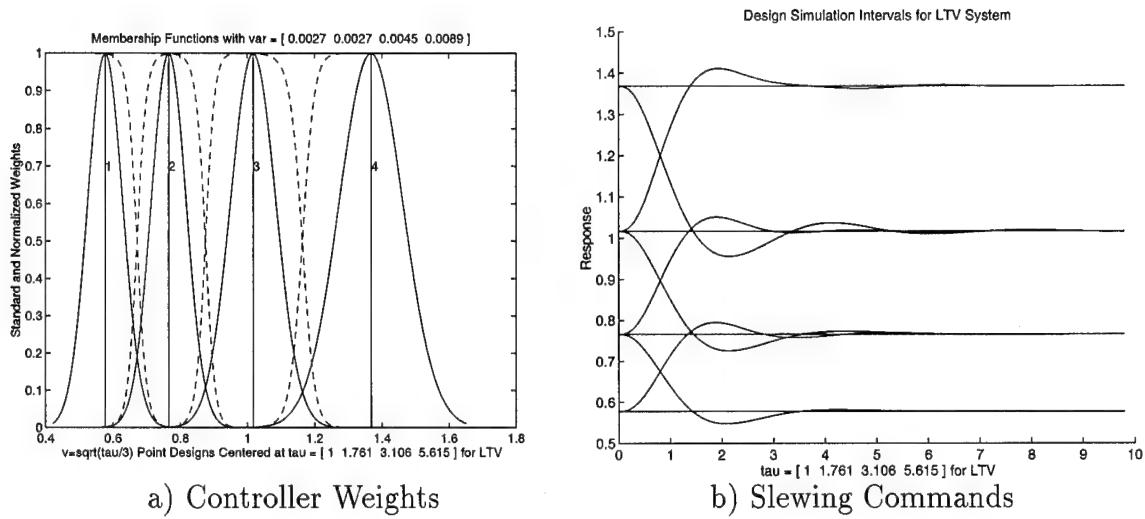
Figure 4.11   Results of Optimization of the LTV System

4.5.1   *Solution for the LTV System.*   The solution for the LTV system yields

$$p^{*^T} = [1 \;\; 1.761 \;\; 3.106 \;\; 5.615] \tag{4.5}$$

$$\sigma^{2^{*^T}} = [2.685e-3 \;\; 2.685e-3 \;\; 4.545e-3 \;\; 8.897e-3] \tag{4.6}$$

where Figure 4.11a shows the controller membership functions and normalized weights. This suggests that the solution to the dual problem requires a maximum of 3 controllers of this type to cover the interval $\tau = [1, 2]$. As shown in the sequel, far greater than 3 point controllers of this type would be required if a conventional scheduling scheme were used. Figure 4.11b demonstrates the slewing between point controllers.

The stability of the controller over $\tau \in [1, 5.615] = \mathsf{P}$ is checked in the same manner as for the nonlinear system. The results in Figure 4.12 indicate a stable system over all of P. Note that for the LTV system, it is the negative commanded steps which are the limiting case.

For the LTV system, a much greater increase in effective cover is observed as shown in Figure 4.13a. There is no overlap between any two regions where individ-
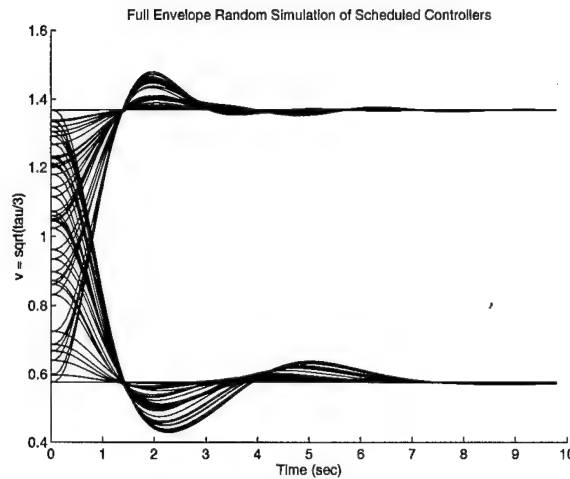
Figure 4.12   Stability Analysis of the Scheduler for the LTV System

ual point controllers provide acceptable tracking performance. Hence, convention scheduling would require greater than the at most three point controllers needed for the fuzzy scheduler. The slewing performance surface in Figure 4.13b depicts a controller with nearly full envelope slewing capability in the LTI sense. This is due to the LTV system being closer to the linear models used to design the point designs.

Notice that in starting from a trim condition near the 4th controller, the performance is less robust for positive steps than from staring near the center of the envelope. For the edge controller, any positive step is away from its valid model and there are no other models that better represent the system. This situation exists for the entire transient. However, from other trim points a combination of models is available during a large portion of the transient. Therefore, when the system reaches the edge of the designated envelope it is in a closer state to the commanded value.

4.5.2   Solution for the LTI System.   Previously it was noted that the LTI system actually poses a different, easier, problem. It is also the most often solved due to the analytic tools available. In cases where the the plant is accurately modeled as an LTI system, this is indeed appropriate and this assumption is made for the example below. However, when the LTI model is obtained by linearizing a nonlinear

4-13

a) Increase in Cover of LTV   b) LTV Performance Surface
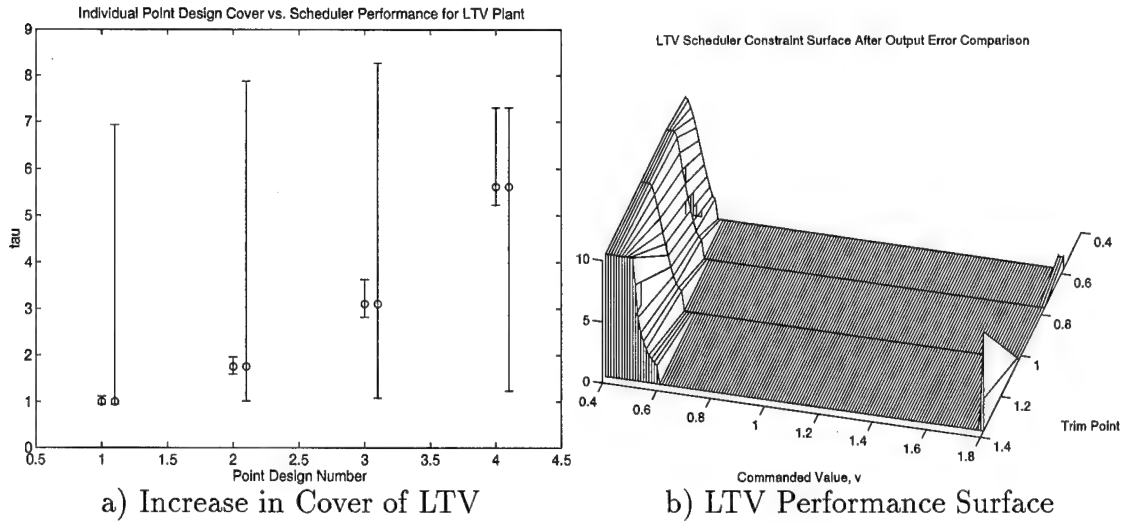
Figure 4.13   Performance of the LTV System

system in order to use these analytic tools, the results can be very misleading. Classically, this is only quantified after the design is complete by simulation, a shortcoming overcome by the synthesis technique of this research. In particular, the nonlinear plant of these examples is such a case where using LTI theory is totally inappropriate as shown by comparing the nonlinear and LTI solutions.

Assuming an LTI truth model for the plant, the solution for $N = 4$ yields

$$p^{*^T} = [1 \quad 1.4 \quad 1.9 \quad 2.5] \tag{4.7}$$

$$\sigma^{2^{*^T}} = [8.8e - 4 \quad 8.8e - 4 \quad 8.8e - 4 \quad 9.4e - 4] \tag{4.8}$$

where Figure 4.14a shows the controller membership functions and normalized weights. This suggests that the solution to the dual problem requires a maximum of 4 controllers of this type to cover the interval $\tau = [1, 2]$. The discussion of cover in the sequel shows that more than four point controllers of this type would be required if a conventional scheduling scheme were used.

4-14

a) Controller Weights

b) Full Envelope Simulations
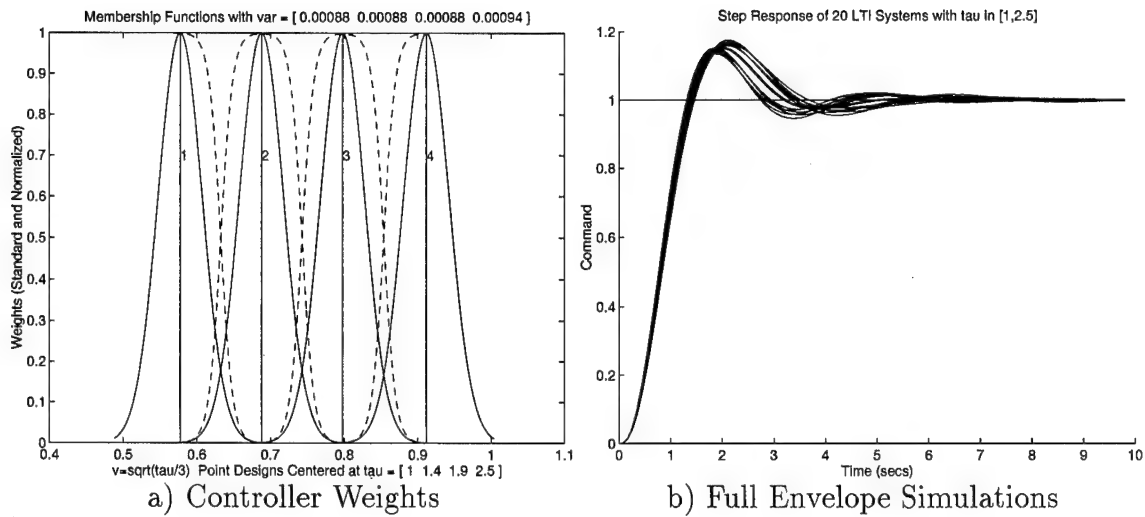
Figure 4.14   Results of Optimization of the LTI System

Figure 4.14b demonstrates the full envelope coverage of the scheduler, in the LTI sense defined below. That is, for the given plant of form

$$P_\tau(s) = \frac{\tau}{s^2 + s + \tau} \tag{4.9}$$

and for any $\tau \in [1, 2.5]$, the scheduler provides acceptable transient response to step inputs. So the optimization yields a *robust* controller with *guaranteed* performance characterizations on the output. These performance guarantees are not just a minimization of an error, which can sometimes be of little true interest. They embed the important specifications on the output and allow for deviations in the response that do not violate them. The performance is also more constrained than in the use of thumb print specifications, and therefore much more useful. Most techniques that promise these results require human intervention to aid in the process, and are not hands off.

The stability of the controller over $\tau \in [1, 2.5] = \mathsf{P}$, may be checked by obtaining the eigenvalues of the system as a function of τ. For any fixed τ, the closed loop system is a deterministic LTI system whose stability is dictated by its eigenvalues,

which can be found by block diagram manipulation or circuit analysis. Define $\alpha(\tau)$ as

$$\alpha(\tau) = \sum_{i=1}^{N} \frac{W_i(\tau)}{p_i} \tag{4.10}$$

where $W_i(\tau)$ is the normalized weight, Eq. (3.1), of the ith controller designed about p_i. Designating the fuzzy scheduler, for fixed τ, as $G_{\text{sch}}(s)$ and combining the N parallel point controllers yields an open-loop system of

$$G_{\text{sch}}(s)P_\tau(s) = \frac{24\tau\left(\alpha^2 s^2 + \alpha s + 1\right)}{s\left(s^2 + 8s + 16\right)\left(s^2 + s + \tau\right)} \tag{4.11}$$

The characteristic equation of the resulting closed-loop system is then

$$s^5 + 9s^4 + (\tau + 24)s^3 + (24\tau\alpha + 8\tau + 16)s^2 + (24\tau\alpha + 16\tau)s + 24\tau \tag{4.12}$$

Since there is no closed form solution for τ in terms of α, a Routhian analysis is performed instead of finding the eigenvalues directly. Clearly the first constraint is $24\tau > 0$. The detailed analysis for general N provides additional constraints, one of which is redundant, yielding the requirements on stability of

$$0 \;<\; \tau \tag{4.13}$$

$$0 \;<\; (1 - 24\alpha)\tau + 200 \tag{4.14}$$

$$0 \;<\; (-72\alpha^2 - 21\alpha + 1)\tau^2 + (309\alpha + 67)\tau + 400 \tag{4.15}$$

$$0 \;<\; (-1728\alpha^3 - 1656\alpha^2 - 240\alpha + 13)\tau^2 +$$
$$(7416\alpha^2 + 8928\alpha + 760)\tau + 9600\alpha - 8000 \tag{4.16}$$

Plots for 3 of the constants and α, as a function of τ, for the design are given in Figure 4.15 from which it is seen that the system is stable for all $\tau \in (0, 11.21)$. Note that α is analytic for all finite N.
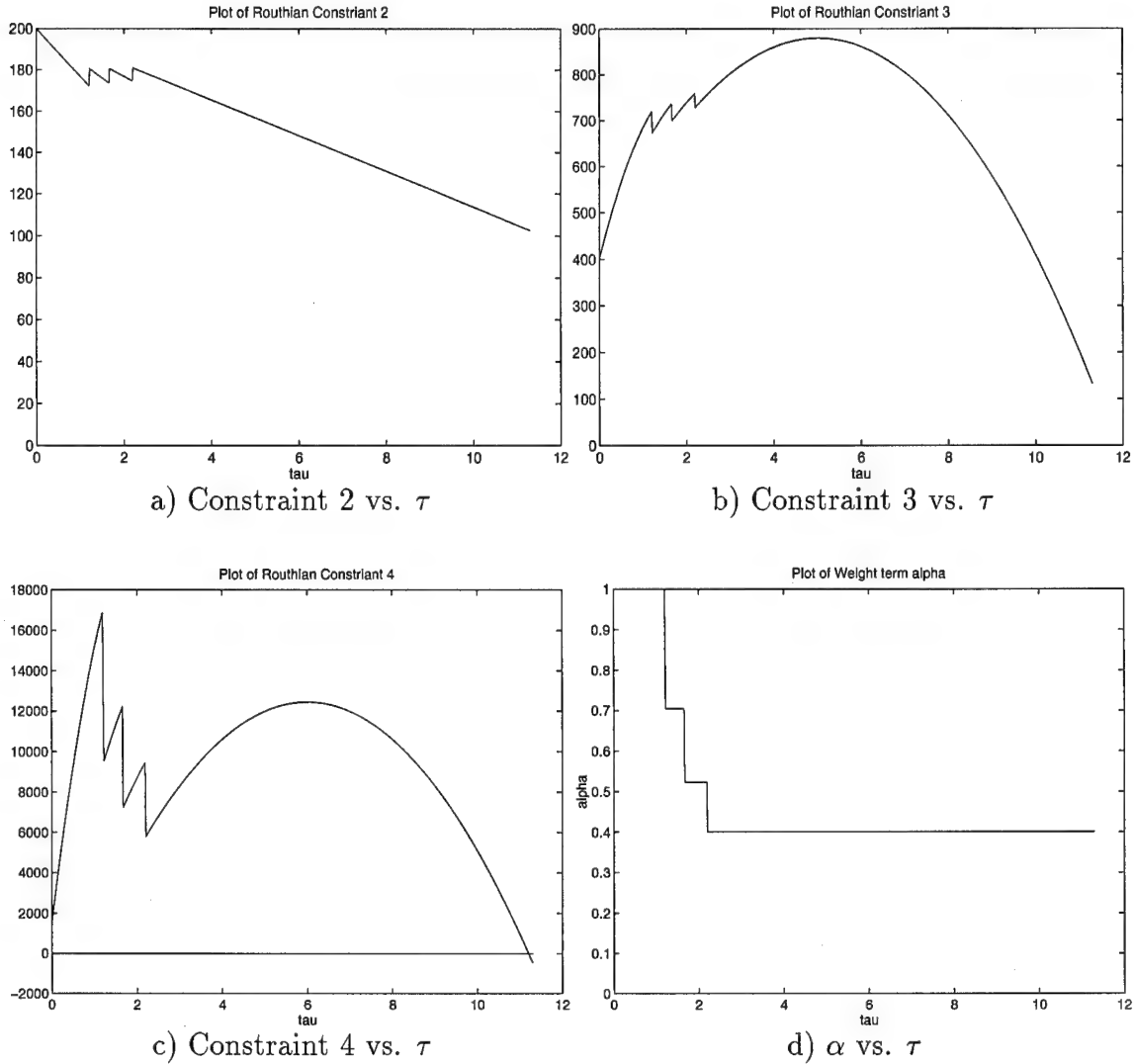
Figure 4.15   Evaluation of Routhian Constraints

For the LTI system, the increase in effective cover is demonstrated by noting that none of the point designs in Figure 4.16 overlap one another. Yet the resulting scheduler covers the interval $\tau \in [0.88, 2.69] \supset \mathsf{P}$. Hence, conventional scheduling would require greater than the at most four point controllers needed for the fuzzy scheduler!
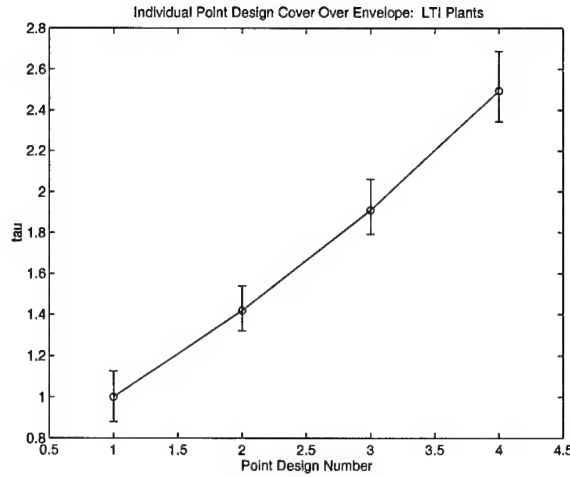
Figure 4.16   Increase of Cover of the LTI System

4.6   C-135 Aircraft Example

An additional example is presented which entails the control of a USAF C-135 transport aircraft. This example is provided to compare the synthesis technique's performance to that of previous designs from the literature [1, 6, 10]. The model used in these designs [12, 13] does not contain sufficient aerodynamic stability derivatives to allow for the full optimization of the point controllers locations. However, the ability to smoothly schedule conventionally designed flight controllers and the robustness achieved demonstrates the wide applicability of the technique.

The goal is to design a coordinated turn controller for the aircraft. This is accomplished by designing the controller such that aileron inputs command bank angle, ϕ, while minimizing the side slip angle β.

4.6.1   The Aircraft Model.   The model presented is the nonlinear three degree-of-freedom lateral directional equations of motion representing a fixed winged aircraft. Only the variables of direct interest are defined here. The notation used is standard for flight control applications and the reader is referred to an aerodynamics text or source materials [7, 12, 13] for a complete description.

4-18

The equations are:

Sum of the forces in the lateral Y direction

$$\dot{v} + U_o r - g\psi \sin\theta_o - g\phi \cos\theta_o = Y_r r + Y_v v + Y_{\dot{v}}\dot{v} + Y_p p + Y_{\delta_a}\delta_a + Y_{\delta_r}\delta_r \quad (4.17)$$

Sum of the rolling moments

$$\dot{p} - \frac{I_{xz}}{I_{xx}}\dot{r} = L_r r + L_v v + L_{\dot{v}}\dot{v} + L_p p + L_{\delta_a}\delta_a + L_{\delta_r}\delta_r \quad (4.18)$$

Sum of the yawing moments

$$\dot{r} - \frac{I_{xz}}{I_{zz}}\dot{p} = N_r r + N_v v + N_{\dot{v}}\dot{v} + N_p p + N_{\delta_a}\delta_a + N_{\delta_r}\delta_r \quad (4.19)$$

Standard small perturbation assumptions are made to linearize the equations. The net result of which are the assumptions

$$\dot{\phi} = p \quad (4.20)$$

$$\dot{\psi} = r \quad (4.21)$$

$$\beta = \frac{v}{U_o} \quad (4.22)$$

Substituting the results of these assumptions into the differential equations and taking the Laplace transform of the resulting equations gives:

$$\begin{bmatrix} \frac{-Y_p s - g\cos\theta_o}{U_o} & s - Y_v & s - \frac{Y_r s}{U_o} - \frac{g\sin\theta_o}{U_o} \\ s^2 - L_p s & -L_\beta & -\frac{I_{xz}}{I_{xx}}s^2 - L_r s \\ -\frac{I_{xz}}{I_{zz}}s^2 - N_p s & -N_{\dot{\beta}}s - N_\beta & s^2 - N_r s \end{bmatrix} \begin{bmatrix} \phi \\ \beta \\ \psi \end{bmatrix} = \begin{bmatrix} 0 & \frac{Y_{\delta_r}}{U_o} \\ L_{\delta_a} & L_{\delta_r} \\ N_{\delta_a} & N_{\delta_r} \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \quad (4.23)$$

It is desired to obtain the state variable representation $\dot{x} = \mathbf{A}x + \mathbf{B}u$. To accomplish this, using the small perturbation definitions, augment the states p (roll rate) and r

(yaw rate) giving the state vector

$$x = [p \; r \; \phi \; \beta \; \psi]^T \qquad . \qquad (4.24)$$

which gives

$$
\begin{bmatrix}
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
1 & -\frac{I_{xz}}{I_{xx}} & 0 & 0 & 0 \\
-\frac{I_{xz}}{I_{zz}} & 1 & 0 & -N_{\dot\beta} & 0
\end{bmatrix} \dot{x} =
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
\frac{Y_p}{U_o} & \frac{Y_r}{U_o}-1 & \frac{g\cos\theta_o}{U_o} & Y_v & \frac{g\sin\theta_o}{U_o} \\
L_p & L_r & 0 & L_\beta & 0 \\
N_p & N_r & 0 & N_\beta & 0
\end{bmatrix} x
$$

$$
+ \begin{bmatrix}
0 & 0 \\
0 & 0 \\
0 & \frac{Y_{\delta_r}}{U_o} \\
L_{\delta_a} & L_{\delta_r} \\
N_{\delta_a} & N_{\delta_r}
\end{bmatrix}
\begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix}
\qquad (4.25)
$$

The system is of the form

$$\mathbf{M}\dot{x} + \mathbf{N}x = \mathbf{D}u \qquad (4.26)$$

defining a five state system with two inputs. Since $\mathbf{M}$ is invertible for $I_{xx}I_{zz} \neq I_{xz}^2$

$$\dot{x} = -\mathbf{M}^{-1}\mathbf{N}x + \mathbf{M}^{-1}\mathbf{D}u = \mathbf{A}x + \mathbf{B}u \qquad (4.27)$$

From Eq. (4.25) the desired LTI plants are obtained as a function of the flight condition (stability derivatives). The value of the parameters for each flight condition are contained in Table 4.1, where the angular measure is degrees.

These flight conditions were chosen in the previous work since they reportedly represent the aircraft well over the flight envelope for lateral dynamics [1, 6, 10].

Flight Condition	Cruise 1	Cruise 2	Power Approach
Altitude (ft)	42,000	25,000	Sea Level
U_o (ft/sec)	726	660	275
Mach No.	0.75	0.65	—
Weight (lbs)	190,000	250,000	165,000
$\bar{q}$ slugs/(ft·sec^2)	141.1562	232.1748	89.9181
Stability Derivative	Flt Cond # 1	Flt Cond # 2	Flt Cond # 3
Y_v	-0.0574	-0.0946	-0.1279
$Y_{\dot{v}}$	0	0	0
Y_p	-1.337	-1.583	-2.294
Y_r	2.621	3.204	4.277
Y_{δ_a}	0	0	0
Y_{δ_r}	16.68	18.73	10.55
L_β	-2.384	-3.109	-1.631
$L_{\dot{\beta}}$	0	0	0
L_p	-0.4695	-0.6381	-0.9074
L_r	0.2341	0.3248	0.5943
L_{δ_a}	0.7227	0.7114	1.433
L_{δ_r}	0.2235	0.3162	0.1223
N_β	0.5089	0.7745	0.2345
$N_{\dot{\beta}}$	0.0110	0	0.0162
N_p	-0.0587	-0.0921	-0.1293
N_r	-0.0927	-0.1506	-0.1503
N_{δ_a}	0.0363	0.0600	0.0403
N_{δ_r}	-0.4965	-0.8278	-0.3305
I_{xx}	3.602×10^6	4.013×10^6	2.813×10^6
I_{zz}	8.648×10^6	8.737×10^6	7.687×10^6
I_{xz}	-7.235×10^5	-2.483×10^5	-2.561×10^5
θ_o (deg)	0	0	-3.0

Table 4.1   C-135 Flight Conditions and Stability Derivatives

4.6.2   Problem Statement.   With the aero data so severely limited, meaningful nonlinear simulations of the aircraft is not possible. Instead, the LTI models of the equations about the flight conditions are used as "truth models". Therefore, the results of the scheduler are interpreted in the LTI sense of the previous section. That is, the performance of the scheduler checks only the robustness, by operating at a flight condition within the available flight envelope but not at a condition that

is a point controller location. To model the aircraft at this non-design point requires using one of the available flight conditions and removing it from point controller consideration. The scheduled point controllers are designed for the other two flight conditions.

It is interesting to note that none of the attempts at this problem in the literature [1, 6, 10] validate their designs in such a rigorous manner. While they make claims of robust designs, their simulations are only performed at points for which the controller was designed. No attempts are made to check performance at off design flight conditions. Claims of a controller's robust performance based solely upon testing against deterministic models for which they were specifically designed are clearly suspect.

Given this situation of scheduling between two controllers, the logical choice of scheduling on one parameter is taken. The question is which parameter of Table 4.1 to use. Although dynamic pressure, $\bar{q}$, does not appear in any equation explicitly, it is common practice in flight control to schedule on *barq* [7]. The flight conditions used to design point controllers are clearly the ones with the minimum and maximum $\bar{q}$, #3 and #2 respectively. The remaining condition (#1), not specifically designed for, is used to test the robustness of the scheduler.

4.6.3 Controller Design. The point controllers are designed in the spirit of classical flight control by use of sequential loop closure [7]. Their design is depicted in Figure 4.17. The first loop is closed using positive feedback of yaw rate,r, with a washout filter in the feedback path to the rudder command. This loop dampens the C-135's severe dutch roll mode. The second loop uses negative feedback of sideslip angle, β, to rudder command. This loop provides aircraft coordination. Finally, the third loop uses negative feedback of the bank angle, ϕ, to aileron command, with a cascade lead compensator in the forward path. An equivalent controller in an appropriate configuration for scheduling is shown in Figure 4.18.
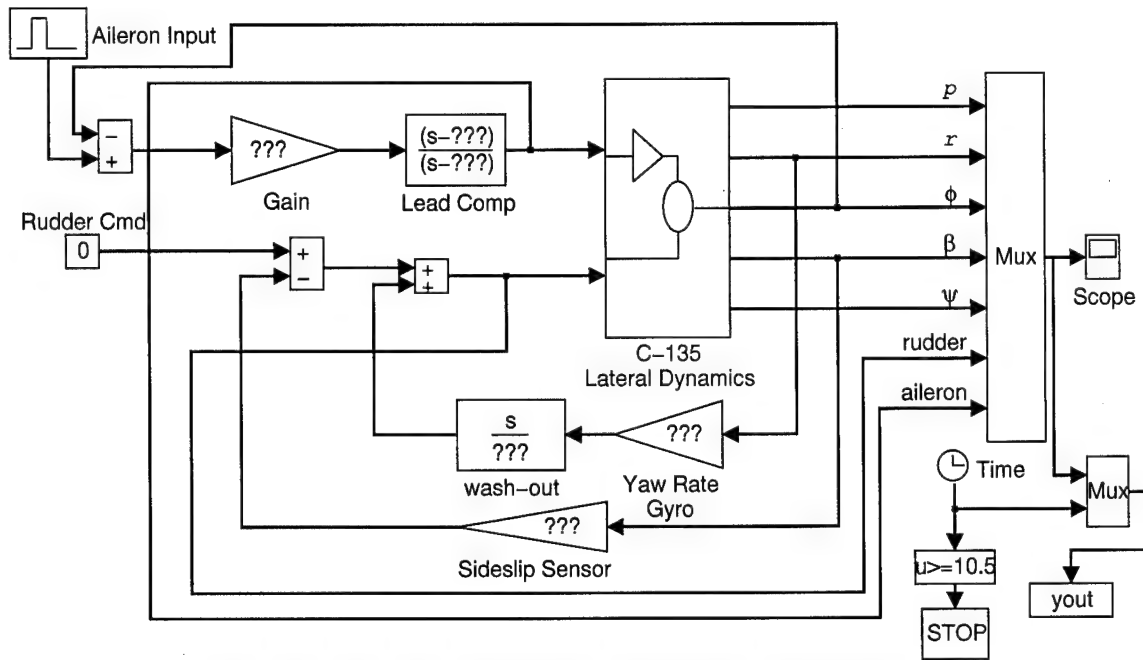
Coordinated Aircraft with Bank Angle (Phi) Controller

Figure 4.17   Classical Flight Controller by Loop Closure

Point Controller for Bank Angle/ Coordination
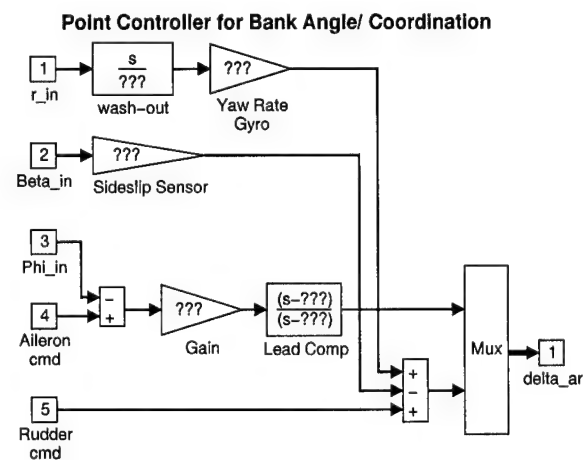
Figure 4.18   Equivalent Form of Point Controller

4-23
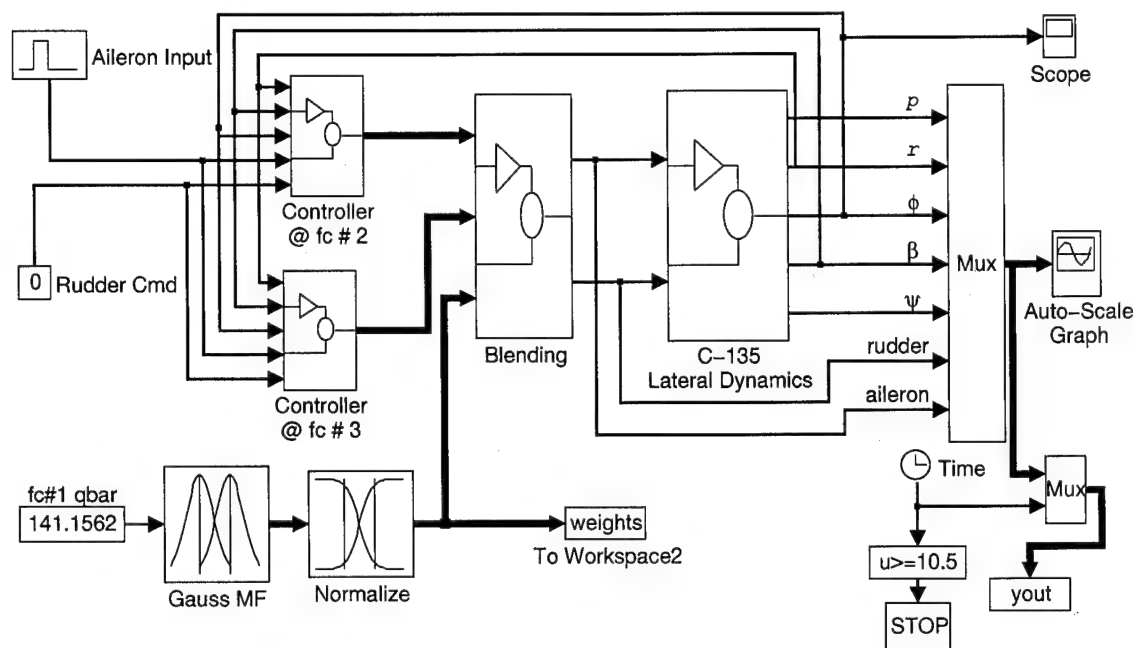
Fuzzy Scheduler of Coordinated Aircraft with Bank Angle (Phi) Controller

Figure 4.19   Fuzzy Scheduler for C-135

The fuzzy scheduler is shown in Figure 4.19. The blending is performed by the components of Figure 4.20 where a cross-membership of 0.001 is used to find the variance parameters.

To evaluate the scheduler's performance, simulations are performed at flight condition #1, which are shown in Figure 4.21 The comparison is made against the response of the two point designs controlling the LTI aircraft at condition #1. The response shows an improvement over the controller for point #3 but noticeable degradation from the controller for point #2, although they may very well be considered acceptable.

This performance is due to the restrictions placed on the scheduling technique by fixing all model locations. The synthesis technique is designed to generate
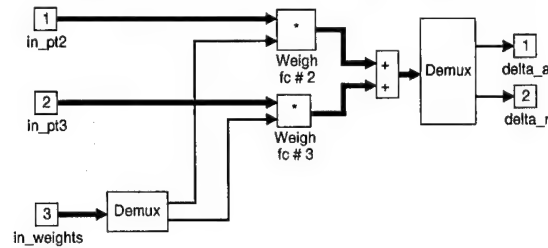
Figure 4.20   Blending of Control Inputs
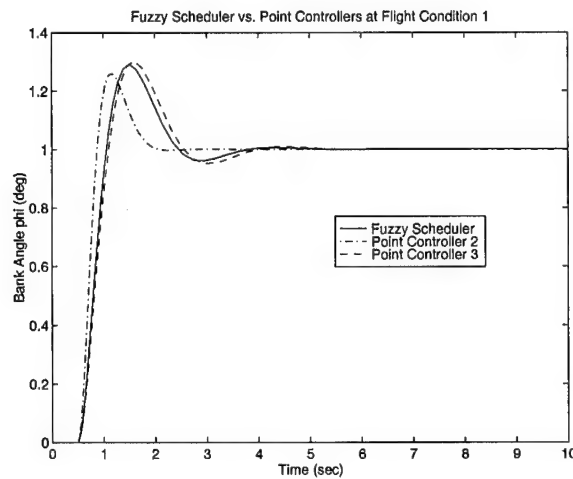
Figure 4.21   Controllers' Performance at Flight Condition #1

membership functions based on point controller locations. The argument of the optimization is the location of these points, so by fixing them the optimization's design freedom is is removed. Fixing these points fixes the MFs, hence contradicting the established paradigm. Thus, the point designs can not be designed independently, a main goal of the synthesis technique. Proceeding with the above restrictions, the use of two fixed points yields symmetric weightings as shown in Figure 4.22.
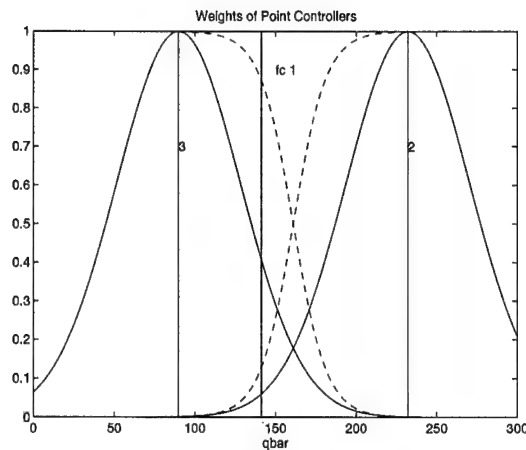
Figure 4.22   Scheduling Surface

The responses of Figure 4.21 shows that the point controller at #2 is robust enough to still perform well at this point. The controller at point #3 has begun to degrade somewhat. However, the location of the test point #1 is closer to flight condition #3 in terms of $\bar{q}$, as shown in Figure 4.22. Hence, controller #3 is weighted more heavily and the scheduler's response resembles it more than that of #2. A more appropriate approach, when the model descriptions are fixed, is to modify the Gaussian membership functions and the resulting normalized weights.

Another suspect area is the scheduling on $\bar{q}$ which is not explicitly in the equations. The obvious conclusion is that the problem at hand is ill posed for the synthesis technique developed in this research. More model information is required to use its full potential. However, the gains of the smooth scheduling provided is evidenced by the blending of the two point controller's characteristics. To match the performance claims of the previous designs [1, 6, 10], one could merely design a third point controller at the last flight condition as they have done. Then rely on the demonstrated smooth scheduling to address off design point performance. However, as pointed out earlier, any conclusion based on such an evaluation is of little value.

4.7 Summary

The fuzzy scheduler developed in Chapter III is used to obtain the full solutions to the problems of Appendix B. The technique is applied to the full nonlinear plant, as well as two simplified models of the plant. In thoroughly exploring the 1-dimensional scheduler and its inherent directionality, this chapter allows one to visualize the gains obtained by the technique. Most notably the increase in controller coverage and the smooth transition between point controllers. It is also pointed out that this sense of directionality allows for a sequential form of optimization to obtain the solution.

A variation on the constraint functional of Chapter III is explored which decreases the amount of over design in the scheduler. The modified constraint embeds both classical figures of merit and a limit on reference output error. The key to using an output error measure as a valid consideration in specifications is establishing a set of acceptable reference outputs, not a single best transient response.

Stability is demonstrated by simulation for nonlinear and LTV plants, and proven in the case of an underlying LTI plant. However, the claim of stability based upon the LTI system when the true system possess such a strong nonlinearity as presented here is obviously of little value.

The design of lateral controller for a C-135 transport aircraft using scheduling is also presented. Lack of necessary model information makes the problem improperly posed to use the synthesis technique to it's fullest. Using LTI models to approximate the aircraft, an evaluation of the scheduler's robustness is performed. The results suggest that in cases where controller locations are fixed a different optimization on the membership function variance parameters is more appropriate. This could act as the degree of freedom that is removed by fixing the controller locations. However, the blending performed by the controller still presents a viable means of switching between point designs.

V. Conclusions and Recommendations

In this research, a novel full envelope controller synthesis technique is developed which directly addresses the control of nonlinear systems with uncertain parameters. This is performed by using Fuzzy Logic to schedule independently designed point controllers over the operational envelope and parameter space of the system's model. These point controllers are synthesized by techniques chosen by the designer, thus allowing an unprecedented amount of design freedom to the controls engineer. By using established control theory for the point controllers, and a Fuzzy Logic scheduler, the resulting nonlinear dynamic controller is able to handle the dynamics of complex systems which can not otherwise be addressed by the direct use of Fuzzy Logic Control alone.

5.1 Conclusions

The previous chapters show that the developed synthesis technique provides a viable solution to nonlinear control problems and uncertain systems with structured parametric uncertainty. The proposed new approach directly addresses those critical areas of FLC and of conventional scheduling design at which the above mentioned paradigms are the weakest.

The use of point controllers based upon established LTI control theory allows for the handling of plants whose complex dynamics require dynamic compensation. This task can not be accomplished with standard FLCs. Thus, the second-order plant example of Chapter III demonstrates non-minimum phase responses which are now controlled appropriately. Nonlinear action is brought in by the FL scheduler. Analytically solving for the membership functions' parameters in the FL scheduler to approximate the Voronoi diagram associated with the point controllers' locations, removes the need to hand tune the MFs or blindly perform an optimization on them, as is usually done in a FLC. Moreover, the normalizing of the membership functions,

along with the concept of cross-membership, ensure adequate and proper cover of the universe of discourse (viz., the plant's operational envelope and the parameter space) even in n-dimensions.

The problem areas for such a task, when attempted by conventional control techniques, are where to place the point controllers in the operational envelope and parameter space, and how/when to either: switch between controllers or schedule parameters in an otherwise fixed controller. In the proposed approach, the drawbacks of mismatched controller/plant energy inherent in switching techniques are now avoided by the smooth transition across a bank of parallel controllers, all under continuous full operation as determined by the fedback plant's state trajectory. This same smooth (analytic over all P) controller weighting surface establishes the scheduling scheme, and hence avoids the ad-hoc nature of conventional scheduling. It is this scheduling on *fast* internal variables, as opposed to slowly drifting parameters in conventional scheduling, which provides the ability to directly handle nonlinear plants. The generation of this normalized weighting surface is performed directly in the n-dimensional state and parameter space in a truly multivariable way, and not by assuming independence of the scheduling parameters. This provides assured control authority over all of P and requires no hand tuning. The placement of the point controllers is a direct outcome of the optimization scheme and is based upon a very meaningful criterion, viz., the system performance while slewing the nonlinear plant. Found in this manner, the spacing between point controllers reflects the relative strengths of the plants nonlinearities and thus requires a fewer number of point controllers than achieved by existing techniques, where uniform spacing over P is employed. It is also noted that an effective increase in cover over the individual point designs is achieved. Therefore, this solution quantifies the required sufficient cover of the point controllers which is unknown a priori.

The result of this research is a systematic methodology resulting in a controller which smoothly transitions the n-dimensional parameter and state space while meet-

ing pre-specified performance requirements. The technique directly handles nonlinear plants with uncertain parameters and does not require "after the design" hand tuning or modifications. It is also demonstrated that the proposed FLC paradigm is directly applicable to uncertain LTI and LTV systems.

5.2 Contributions

In the development of this synthesis technique, several unique contributions to the field of control are introduced:

1. Full envelope control of dynamic multivariable nonlinear plants.

2. Directly design for plant nonlinearities and parameter variation using conventional LTI control design tools embedded in a Fuzzy Logic paradigm. The solution is obtained in a systematic manner which does not require modifications or hand tuning when evaluated against the uncertain nonlinear plant.

3. Physically meaningful method of analytically generating true multivariable n-dimensional membership functions.

4. Introduced novel computational geometry concepts of Voronoi/Delaunay diagrams into control theory, to allow for systematic multi-dimensional (multivariable) scheduling. These concepts are also applicable, and in fact are appropriate, to other multivariable scheduling or switching techniques.

5. Analytic means of quantifying point design cover and switching surface generation by approximating multi-dimensional Voronoi diagrams with same dimension multivariable Gaussian membership functions.

6. Formal definition of cross-membership provides a means of dealing with a set of non uniformly distributed multi-dimensional membership functions without compact support.

7. Solutions obtained by the developed technique extend the coverage of individual point designs, therefore requiring less point controllers than required by existing switching/scheduling schemes.

8. Removed FL requirement of using composite clauses in the antecedent by means of multivariate fuzzy AND operator.

9. Accomplished scheduling of a bank of dynamic controllers that avoids mismatched energy between controller and plant, that provides smooth transitions between point controllers.

10. A current shortfall in FLC addressed: Use of LTI control tools instead of direct fuzzy inference allows for the handling of systems possessing complex dynamics. Thus, dynamic compensation is ported into FLC.

11. Optimization maximizes cover of convex hull (operational envelope) while meeting output response specifications. Thus, the solution to the minimum number of point controllers required to cover a specified envelope is readily obtained.

12. Posed a performance functional combining figures of merit and output error with regards to various reference signals. This gives the output error metric a much larger class of applications than that of model following.

5.3 Recommendations for Further Study

Given the fuzzy controller scheduler architecture developed in this research, several areas are suggested as topics of further research. They fall into two categories: those which improve the ability to obtain a solution, and those which extend the work presented here.

Improving the convergence to a solution would perhaps most benefit from changes in the optimization routine used to obtain the solution. The technique in Chapter III is posed such that any routine which can solve a constrained optimization

(minimization) could be used. In particular, the multi-dimensional example of this research uses the Sequential Quadratic Programming routine from MATLAB® 's *Optimization Toolbox* [28]. Modifications such as two-sided finite differences, to estimate gradients, should aid in progress towards a solution. Also, additional optimization algorithms should be investigated.

The ability to include the second step of evaluating the constraint $C(y(t))$ in dimensions greater than one would aid the solution in that its effect is to ease the constraints. This area could also be improved by refining of the feature extracting functionals and the specification vector. In particular, the relationships regarding ratios of oscillations in the output response are conservative. In its current form, the ratios specification properly identifies a class of responses as acceptable and properly penalizes another class that is hard to identify as unacceptable by conventional time specifications. However, this spec does effectively reject certain responses which would probably be judged acceptable by observation; thus, over constraining the system and increasing the difficulty in obtaining a numerical solution.

Finally, there is a need for employing more efficient computational geometry algorithms in calculating the Voronoi and Delaunay diagrams, especially as the scheduling dimension increases.

To extend the technique past its current development, the two main areas of analysis and application are considered. For analysis, the question of system stability could be addressed. In this research, stability is addressed by extensive simulation over the operational envelope. One could instead attempt a general stability analysis of either the general fuzzy controller structure, or a stability proof of a specific controller after the design has been completed.

In the area of application, the demonstration of full MIMO control by means of solving several equivalent MISO subproblems can now be addressed. The design of such a controller would demonstrate the full power of the synthesis technique developed in this research.

5.4 Summary

This research has made major contributions to full envelope nonlinear control and dynamic Fuzzy Logic control. Presented is a systematic means of developing a nonlinear controller which accommodates the system model's nonlinearities and parameter variation. The technique seeks to maximize the coverage of the operational envelope while guaranteeing a pre-specified transient performance and smooth transitions across the envelope. The resulting controller does not require on-line adaptation, estimation, prediction or model identification to achieve this objective. Complex dynamics are handled by relying on conventional control theory for the point designs and avoiding the use the system's error state for the fuzzy inference engine. A meaningful analytic solution of the membership function variance allows the optimization to yield the location of point designs: both quantifying the controller's coverage, and eliminating the need of extensive hand tuning of these parameters.

The above is a significant contribution to the field even in the case of scheduling one parameter for a nonlinear SISO plant. Beyond this, the geometric primitives used in the solution all have higher dimensional interpretations (convex hull, ellipsoid, Voronoi/Delaunay diagrams) which allow for a direct generalization to scheduling on n-dimensions including uncertainty due to nonlinearities and parameter variation. This is all achieved in a direct systematic manner which requires no hand tuning of multi-dimensional membership functions. Since many MIMO controller design techniques are accomplished by solving several MISO problems, this work bridges the gap to full envelope control of MIMO nonlinear systems with parameter variation.

Appendix A. Fuzzy Identification

There are two methods to obtain models of physical systems:

1. From first principles, using the physical sciences and their mathematical descriptions.

2. Take a statistical/ black box approach. where one fits a model to empirically obtained input/output data. Fuzzy Logic can play a role in this approach to modeling.

In the main research, the emphasis is for the application of Fuzzy Logic concepts in the case when approach 1 is taken. In contrast, the discussion of this appendix examines approach 2 and is included for completeness. That is, in this appendix, the assumption is made that due to a lack of confidence in a mathematical model of the plant but the availability of empirical data, one may attempt identification based solely upon the input/output pairs.

To gain insight from the underlying fuzzy inference engine, system identification using Fuzzy Logic modeling is investigated in two areas. The first entails the representation of an unknown input/output mapping by a fuzzy inference engine, using input/output data. The second addresses the class of problems where the fuzzy system's structure is known and one is concerned with the identification of the underlying fuzzy rules from input/output data. Examples, using polynomial models and a logical XOR device, respectively illustrate the two proposed fuzzy logic modeling/identification paradigms.

The direct application of Fuzzy Logic as a controller requires imbedding rules in a fuzzy inference engine. However, in the case of complex systems, the fuzzy rules are not so easy to come by. Thus, there is a need for system (rule) identification as an inherent part of the FLC based design process. In particular, when the resultant

controller description is in the form of rules in a fuzzy inference engine, the need for fuzzy identification becomes obvious.

A.1 Fuzzy Logic ID Paradigm

Much of the work in this area assumes total ignorance of the plant and attempts identification based purely on plant input/output data, a data driven approach. Such cases can be handled by using Neural Networks which are being trained to respond similarly to the actual plant, and therefore may be used as the implicit repository of the underlying rule base [4, 18]. An alternative approach assumes a generic form of parametrically represented membership functions and performs an optimization on these parameters to obtain a good fit to the input/output data. The result provides a fuzzy rule set which best fits the provided data, given the form of the membership functions [19, 22].

In this appendix [20] a class of systems is considered in which expertise on the plant's operation is available. That is, much is known about the plant in question, but mathematical equations are not reliable due to either unmodeled dynamics or parameter uncertainties. This class of problems includes those in which experienced operators can control the system but can not verbalize the rule set. In particular, problems are discussed for which sufficient prior information exists to stipulate the following:

1. The universe of discourse

2. The number of fuzzy input/output variables required

3. Fuzzy variable values (i.e. small, medium or large)

4. Appropriate membership functions

To complete the hypotheses, fuzzification (min/max, product, etc.) and defuzzification (centroid, etc.) algorithms must be specified. Finally, the "plant" being

modeled can represent an actual physical plant or a human operator controlling the physical plant. In the case of modeling the plant's controller, one is tying to emulate the experienced operator's rules in the FLC.

A.2 Identification Concept

The set R is defined as the finite set of all feasible rule sets (which adhere to the hypotheses). This set is well defined and one can perform an exhaustive search to find the optimal elemental rule set $r^* \in$ R which best represents the system of interest. In line with the classical system ID paradigm, the mean-squared output error metric of the rule's action on the input data set, $r(x_i)$, versus actual output data, y_i, is used. In the scalar case with input/output data (x, y), the optimal rule set $r^* \in$ R satisfies:

$$r^* = \arg \min_{r \in R} \frac{1}{n} \sum_{i=1}^{n} \left(r(x_i) - y_i \right)^2 \tag{A.1}$$

Obviously, analysis cannot be brought to bear on the solution of the above discrete optimization problem. As the number of fuzzy variables increase, minimization by exhaustive search over R suffers from a combinatorial explosion. However, it allows for a well posed problem in which an optimum exists. Suboptimal solutions are provided by genetic algorithms [11, 19] or other numeric optimization methods. The proposed system ID paradigm is demonstrated in two examples: 1) By identifying noise corrupted polynomial input/output mappings and 2) By identifying a logic XOR gate.

A.3 Polynomials

A system with the following fuzzy hypotheses, obtained from experience, is used for this example.

A-3
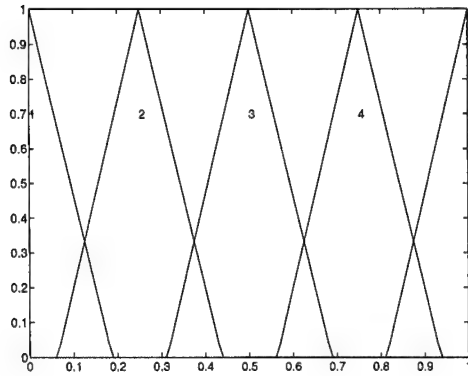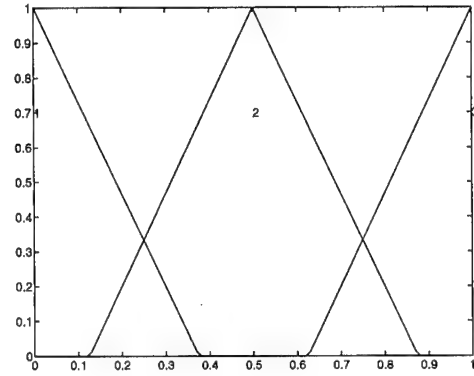
a) 5 Fuzzy Inputs   b) 3 Fuzzy Outputs

Figure A.1   Triangular Fuzzy Variables for Input/Output

The universe of discourse is scaled to the interval [0,1]. Five input fuzzy variables (FV) and three output FVs are used. Their linguistic labels are 1-5 and 1-3 respectively where triangular membership functions represent these variables. These are illustrated in Figures A.1a and A.1b. The fuzzification and defuzzification algorithms are the min/max and centroid methods, respectively.

A single rule maps an input to an output, e.g.,

Example Rule $\mathcal{R}$:   **IF** *input* is 2,   **THEN** *output* is 1.

A rule set is a set of rules mapping each input FV to a unique output FV. So for n input FVs and m output FVs, the set R contains m^n possible rule sets (combinations of rules). Results for data generated from noise free 1st and 2nd order polynomials are shown in Figures A.2a and A.2b respectively. The identification is performed in MATLAB® [29]. In the first example, R contains 243 possible rule sets with five rules each; hence, the second order fit requires $5^5 = 3125$ different possibilities to be investigated. In Figure A.2a the truth model $y = x$ is compared to the output of the best rule set as applied to the universe of discourse [0,1]. In Figure A.2b, the true values are the discrete points, while the curve is the output of

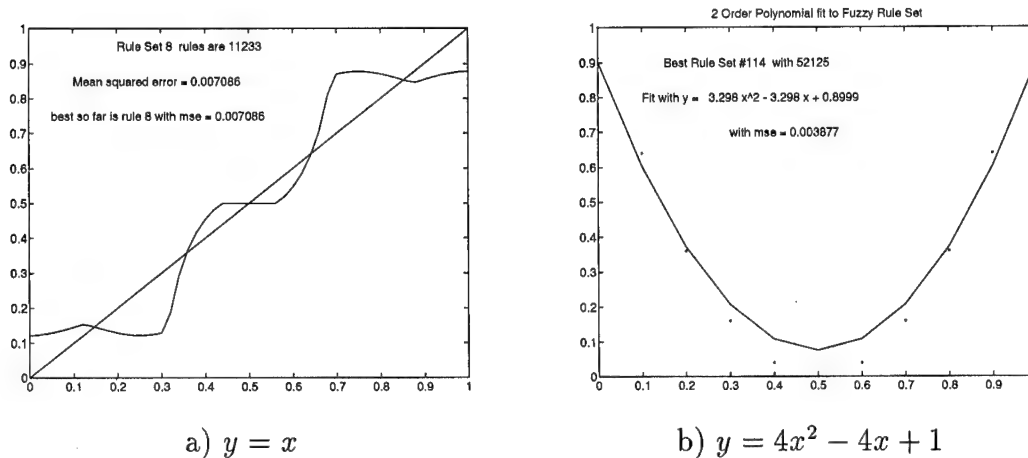a) $y = x$ b) $y = 4x^2 - 4x + 1$
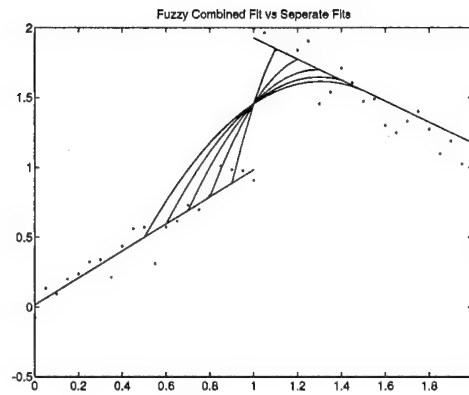
Figure A.2   Fuzzy Fit of Polynomials

the best fuzzy rule set. In both figures the fuzzification is performed by the min/max operators. The exhaustive search may be greatly reduced by first eliminating any of the possible rules that are never activated by a point in the data set.

Of course, there are many ways to fit polynomials, but this validates the usefulness of the above outlined search.

A great advantage of a fuzzy modeled system becomes evident when the finite, and hence incomplete, data set appears to be discontinuous, while one has every reason to believe that the underlying true system is continuous. An example of such data can be seen in Figure A.3a along with two attempts at fitting the data. To avoid the dangers of over-fitting data with higher order polynomials, the solid lines represent 1$^{\text{st}}$ order fits in a least-squares sense. The one using the entire domain [0,2] is continuous, but yields a poor fit to the data. When the interval is broken into [0,1] and [1,2] excellent fits are obtained, but this results in a discontinuous system. However, if one defines overlapping membership functions, in order to combine the discontinuous fits, a continuous transition is obtained along the entire [0,2] interval. Figure A.3b depicts this for several values of overlap from 5% to 40%. Again the

A-5
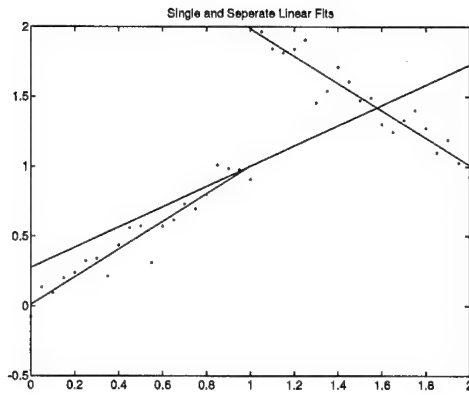
a) Single and Separate Fits   b) Fuzzy vs. Discontinuous Fit

Figure A.3   Fuzzy Blending as a Function of Overlap

fuzzification uses min/max. A smooth blending of the two separate fits is obtained when a "product rule" for combination and analytic membership functions are used.

A.4   XOR Gate Plant Example

A classical system (in Neural Networks) is considered with two inputs and one output which is believed to be a noise corrupted binary device. Because of this it is decided to model each input and the output with two FVs, "small" and "big". This establishes a truth table which shows that there are only 16 possible rule sets, of 4 rules each, to search. Nearly all measurements are in the interval [-1,2], so this is used as the universe of discourse. This time, Gaussian membership functions centered at 0 and 1 are chosen to represent the fuzzy variables "small" and "big" respectively. The fuzzy AND operator is implemented with the product rule.

$$\mu(x) = \mu_s(x)\mu_b(x) = e^{\frac{-x^2}{2\sigma_s^2}} e^{\frac{-(x-1)^2}{2\sigma_b^2}}$$

Defuzzification is again accomplished by the center of area method.

Inputs		16 Possible outputs															
1	2	1	2	3	4	5	6	**7**	8	9	10	11	12	13	14	15	16
S	S	S	S	S	S	S	S	**S**	S	B	B	B	B	B	B	B	B
S	B	S	S	S	S	B	B	**B**	B	S	S	S	S	B	B	B	B
B	S	S	S	B	B	S	S	**B**	B	S	S	B	B	S	S	B	B
B	B	S	B	S	B	S	B	**S**	B	S	B	S	B	S	B	S	B

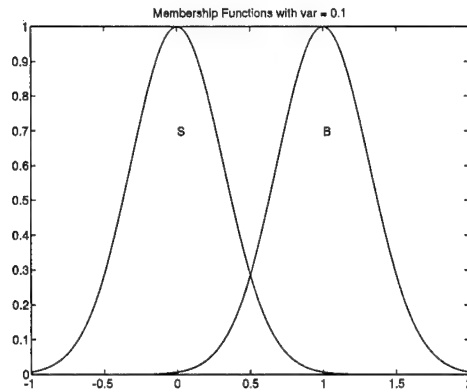Table A.1   All Possible Rule Sets for 2-input 1-output Binary Device

Figure A.4   Gaussian Fuzzy Variables for all Channels

The input/output truth data set for identification is generated using a fuzzy XOR gate as the truth model (rule 7 in Table A.1) from the above membership functions $\mu_s(x)$ and $\mu_b(x)$ with variance parameter $\sigma_s^2 = \sigma_b^2 = \sigma^2 = 0.1$. These membership functions are shown in Figure A.4. The inputs are fed to every possible rule set and the output is compared to that obtained from the truth model (the original XOR). During an actual search one would not know the variance either, so that the problem entails both structural identification (rule set) and parameter identification (σ).

An exhaustive search with fixed σ is made to find the minimum error, and hence best fit. However, σ is unknown and hence R is infinite. The problem is not
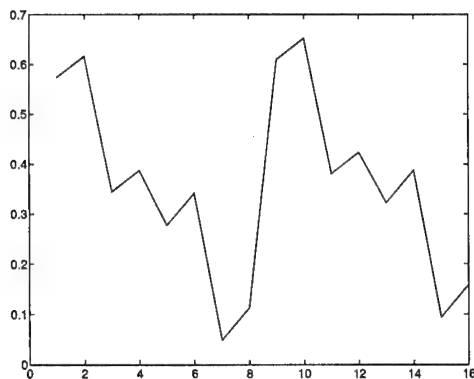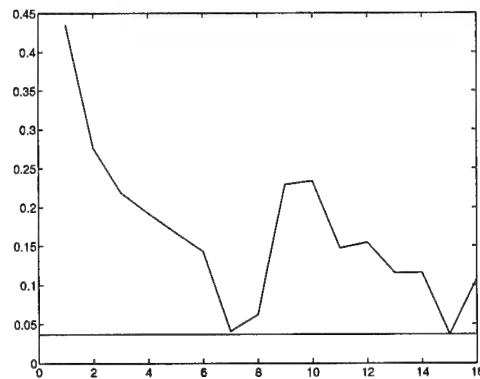
a) Correct Rule, $\sigma^2 = 0.1$ b) Mis-Identification, $\sigma^2 = 0.5$

Figure A.5   Identification Results: Error vs. Rule Set

convex with respect to the rule sets in general, so finding the global minimum is no longer assured by a non-exhaustive (finite) search.

If there were a guarantee that for any fixed σ^2 one could identify the correct rule set the process would indeed be straight forward. One would fix σ^2 and identify the true rule set (truth table), then one would merely minimize with respect to σ^2. However, this may not be the case, as shown in Figures A.5a and A.5b, which show different identified rule sets for different σ and where rule set 7 represents a logical XOR gate, see Table A.1.

Figure A.5a shows the correct rule set (7) identified for $\sigma^2 = 0.1$. Now, let $\sigma^2 = 0.5$ in all 16 rule sets for which the fit of the data is to be performed (Note: the I/O data is generated for the nominal σ^2). Figure A.5b shows that rule 15 (with $\sigma^2 = 0.5$) now has the smallest output error, and hence the incorrect rule set has been identified.

The reason behind the above mis-identification plagues all identification techniques that rely solely on output error minimization. Thus, these system identification techniques require "good" excitation in order to work. These techniques are

A-8
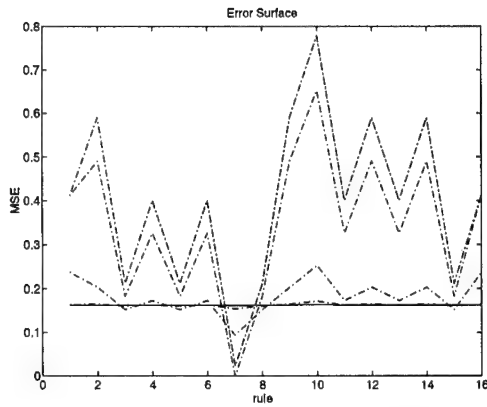
a) Error Surface vs. $\sigma^2 \in [0.001, 100]$ b) Correct ID, $\sigma^2 = 100$
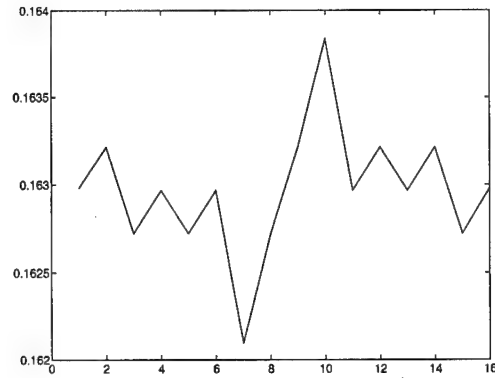
Figure A.6   Performance for Spanning Input/Output Data

dependent upon how well the data represents or spans the input and output spaces of the system. The input data for Figures A.5a and A.5b are randomly generated from the universe of discourse and then run through the fuzzy XOR. This is equivalent to taking passive noisy input/output measurements to use in identification. Unfortunately, the data points are not spread about (do not cover) the entire input space. Thus they do not excite the entire dynamic range of the system and actually misrepresent the plant. Notice from Table A.1 that spanning, or covering the input space requires sufficient points from the four input antecedents (s,s), (s,b), (b,s), (b,b). When input pairs are purposely chosen in an attempt to span the input space over [-1,2] error surfaces similar to Figures A.6a and A.6b are obtained. That is, for any σ, rule set seven is identified, and then minimizing the output error with respect to the variance parameter gives $\sigma^2 = 0.1$.

Hence, as is well known in classical system identification, good excitation is a prerequisite for successful identification. If the data points available do not appear to span the spaces very well, heuristically one should start with a very small σ. Then, for the fixed σ find the best rule set and its goodness of fit. Increase σ and repeat

A-9

until the optimal rule set changes or until σ increases beyond a reasonable value for the system. For each rule set identified as optimal for some σ, minimize with respect to a varying σ. Choose the best global fit as the "model". It is believed that the identified rule set corresponding to the smallest σ is the global optimum with respect to the hypothesized membership functions, but no proof is provided. This has been observed in all simulations performed. The results are found to be invariant with respect to varying the variance parameter as well as the rule set used in generating the truth data.

A.5 Summary

Rule identification clearly plays a central role in the application of fuzzy logic control, where a human expert's plant and/or control knowledge needs to be captured. Although there are techniques which provide these rules merely from input/output data, they often overlook prior and/or side information which can improve their performance. One should strive to encode all available knowledge of the system into the fuzzy model at the onset. This information adds structure to the problem, allowing a more efficient and correct solution. If properly posed, the problem may lend itself to an exhaustive search, guaranteeing an optimal solution. The use of such a search in some illustrative examples is presented. Identification experiments are performed, where the quality of the ID algorithm is validated, since the underlying truth model (which encoded the data) is available in the experiment. If an exhaustive search for optimization is too expensive, the use of genetic algorithms is suggested, as they very naturally fit the proposed paradigm [11, 19].

The requirement for good excitation in system identification is also shown in this appendix. Without it, the results can be misleading and have dramatic effects when the fuzzy model is used away from its nominal operating point where the ID experiment is performed. Indeed, a word of caution that applies to all ID work is in order: Obtaining a small output error is no prerequisite to correct ID,

except in the case where the ID experiment is performed under conditions of "good" excitation. This reconfirms what is known from classical system identification, where the following dictum should be adhered to: If the excitation is poor, don't ID. A challenging problem in its own right is the independent and (input/output) data driven determination of the excitation level.

The above work demonstrates the blending capabilities of a fuzzy supervisor in Figure A.3b, where the "point designs" are the two separate linear fits to the data. The result is continuous when the normal fuzzy operations (min/max) are used. However, in the blending of dynamic controllers, one desires a smooth transition between point designs. Hence, the choice of analytic membership functions is suggested.

Appendix B. Experiments

Experimentation is valuable to the engineer and mathematician, often providing new insights and confirmation or denial of mathematical formulations and previously held opinions. Experimentation is most valuable and empowering when used to disprove previously held truths, as is often the case in exploratory research. Since the goal in this research is to develop new theory, these experiments can be used to guide the way and eliminate paths that would not be as successful. It is desired to develop a technique which uses Fuzzy Logic to provide full envelope control for nonlinear MIMO systems with complex dynamics.

B.1   Nonlinear Plant Formulation

Consider the general control problem as it applies to the nonlinear "bare" plant for which the controller is to be designed. Assume that a model of the plant, albeit complex, may be represented in the form

$$\dot{x}_f = f(x_f, u_f; p)$$

where the mapping f is sufficiently smooth in all of its arguments to use a first order Taylor's series expansion where $x_f \in \mathbb{R}^n$ and $u_f \in \mathbb{R}^m$ are the state and control vectors, respectively. The subscript f indicates full states as opposed to perturbation states, and for most physical systems $n \geq m$. To indicate the dependency on current operating conditions, an additional parameters' vector $p \in \mathsf{P}$ is also included.

Complete state observation is stipulated by assuming the system's state measurement x_f is available for feedback. As noted previously, components of the state vector x_f can be both fuzzy or crisp. If the entire state vector is considered fuzzy, x_f's crisp components mandate a "singular" covariance matrix $\mathbf{R}$.

Consider the operating point/equilibrium condition, or in aeronautical termi-
nology, trim condition $(\bar{x}, \bar{u})$. Since by definition this is a static equilibrium point,
$\dot{x}_f = 0$ giving

$$f(\bar{x}, \bar{u}; p) = 0 \qquad\qquad (B.1)$$

Linearizing about the specified trim condition $(\bar{x}, \bar{u})$, yields the linear time-invariant
(LTI) plant

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$

where x and u are the perturbation variables

$$x = x_f - \bar{x}, \qquad u = u_f - \bar{u}$$

and the matrices $\mathbf{A}$ and $\mathbf{B}$ are the Jacobians of the nonlinear mapping $f(x_f, u_f)$
evaluated at the relevant trim condition $(\bar{x}, \bar{u})$, e.g.

$$\mathbf{A} = \left. \frac{\partial f(x_f, u_f)}{\partial x_f} \right|_{(\bar{x}, \bar{u})}, \qquad \mathbf{B} = \left. \frac{\partial f(x_f, u_f)}{\partial u_f} \right|_{(\bar{x}, \bar{u})}$$

Clearly,

$$\mathbf{A} = \mathbf{A}(\bar{x}, \bar{u}; p), \qquad \mathbf{B} = \mathbf{B}(\bar{x}, \bar{u}; p)$$

In most cases of interest Eq. (B.1) has a non-unique solution $(\bar{x}, \bar{u})$ generating,
a possibly infinite, set of equilibria points, each yielding a different $\mathbf{A}$ and $\mathbf{B}$. The
above LTI perturbation model represents the underlying nonlinear plant as long as
the higher order terms of the Taylor expansion are sufficiently small. That is, the
LTI model is "good enough" in some region about the trim point $(\bar{x}, \bar{u})$. Since the
objective is to design for the entire range of operation, this will require a number
of trim conditions of interest, say N, that "cover" the operational envelope in $\mathbb{R}^n$.

Denote the N trim conditions of interest by $(\bar{x}_1, \bar{u}_1), \ldots, (\bar{x}_N, \bar{u}_N)$. Hence, N LTI plants $(\mathbf{A}_1, \mathbf{B}_1), \ldots, (\mathbf{A}_N, \mathbf{B}_N)$ are obtained, where

$$\mathbf{A}_i = \mathbf{A}(\bar{x}_i, \bar{u}_i; p), \quad \mathbf{B}_i = \mathbf{B}(\bar{x}_i, \bar{u}_i; p)$$

Thus, the perturbed state and control vectors satisfy

$$\dot{x} = \mathbf{A}_i x + \mathbf{B}_i u \qquad \forall \, i = 1, 2, \ldots, N \tag{B.2}$$

Returning momentarily to the trim condition Eq. (B.1), observe that, for a given parameter vector p, it entails n equations in the $n + m$ unknowns $\bar{x} \in \mathbb{R}^n$ and $\bar{u} \in \mathbb{R}^m$. Hence, under relatively mild conditions, elaborated on shortly, the control $\bar{u}_i$ required for trim is determined by m components of the equilibrium state $\bar{x}_i$; this m-dimensional subspace of the trim state vector $\bar{x}_i$ thus defines an m-dimensional projection of the operational envelope. In other words, there exists a function $f_t :$ $\mathbb{R}^m \to \mathbb{R}^m$, such that

$$\bar{u}_i = f_t(\bar{x}_i^{(m)}; p)$$

where the $x_f^{(m)} \in \mathbb{R}^m$ vector consists of the above mentioned m components of the state. Moreover, there also exists a function $g_t : \mathbb{R}^m \to \mathbb{R}^{n-m}$, such that

$$\bar{x}_i^{(n-m)} = g_t(\bar{x}_i^{(m)}; p)$$

where the $n - m$ dimensional vector $x_f^{(n-m)}$ consists of the remaining $n - m$ components of the state vector x_f, i.e.,

$$x_f = [x_f^{(m)'}, \; x_f^{(n-m)'}]'$$

Thus, the (nonlinear) trim equation yields the trim control $\bar{u}_i$ and $n-m$ components $\bar{x}_i^{(n-m)}$ of the trim state, as a function of the remaining m components $\bar{x}_i^{(m)}$ of the trim state. Indeed, the m-dimensional vector of *independent* trim variables $\bar{x}_i^{(m)}$ renders the operational envelope m-dimensional. This will result in exploring an m-dimensional projection of the operational envelope, that is specified by the m state components of $x_f^{(m)}$. Hence, the N m-dimensional trim "states" $\bar{x}_i^{(m)}$, $i = 1, 2, \ldots, N$ must properly "cover" the m-dimensional operational envelope.

Furthermore, the above functions f_t and g_t are continuous, provided the below Jacobians are non-singular [34].

$$\text{rank } [\mathbf{B}(\bar{x}_i, \bar{u}_i; p)] = m \qquad \forall\, i = 1, \ldots, N$$

B.2   Two State Nonlinear Example

In order to probe the applicability of the desired technique, experiments are performed with a strongly nonlinear two-state SISO plant with parametric uncertainty taken from an investigation into intelligent control [25, 26]. The model is

$$\dot{x}_{f_1} = -x_{f_1} + ax_{f_2} \quad a \in [0.5, 1.5] \tag{B.3}$$

$$\dot{x}_{f_2} = -x_{f_1}^3 + u_f^3 \tag{B.4}$$

$$y_f = x_{f_1} \tag{B.5}$$

When addressing only uncertainty due to linearization, a is set to 1. This is a particularly difficult nonlinear plant. The nonlinearity does not constitute a small perturbation of an otherwise linear plant. The cubic nonlinearity is particularly severe. Furthermore, the inherently nonlinear dynamics are *faster* than the dynamics of the linearized plant, to be shown shortly. That is, the trim states move faster than the linearized dynamics. This means that scheduling would be particularly difficult

as one would have to schedule on *fast variables*. In control design practice, fast variables are used as primary feedback and scheduling is performed on *slow variables* [34]. This is particularly true as $x_f \longrightarrow 0$ where the plant becomes uncontrollable. Also, recall that in the case of a cubic nonlinearity, it is impossible to predict the stability of the nonlinear plant from the linearized dynamics [26].

The Mathworks, Inc. products [29] MATLAB® (numeric computation and visualization) and SIMULINK® (dynamic simulation environment) are used throughout these examples.

At a static equilibrium point one has $\dot{x}_f = [\dot{x}_{f_1}, \dot{x}_{f_2}]' = 0$. Thus, using the previously defined notation and solving for the trim condition, the model yields $\bar{x}_1 = a\bar{x}_2 = \bar{u}$. This gives the perturbation equations as

$$
\dot{x} = \left. \begin{bmatrix} -1 & a \\ -3x_{f_1}^2 & 0 \end{bmatrix} \right|_{x_{f_1}=\bar{x}_1} x + \left. \begin{bmatrix} 0 \\ -3u_f^2 \end{bmatrix} \right|_{u_f=\bar{u}} u
$$

$$
y = x_1
$$

Defining $\tau = 3\bar{x}_1^2 = 3\bar{u}^2$ and evaluating the above matrices at the equilibrium point gives the LTI system as

$$
\dot{x}_1 = -x_1 + ax_2 \tag{B.6}
$$

$$
\dot{x}_2 = -\tau x_1 + \tau u \tag{B.7}
$$

$$
y = x_1 \tag{B.8}
$$

This plant is used in this appendix to perform a preliminary exploration into the effects of nonlinearities and parameter variation on the scheduling of LTI controllers. This demonstrates the non-trivial nature of the problem at hand, and points to many concerns which must be addressed by the final synthesis technique. The

plant's use in Chapter IV of the main document allows for validation of the final technique.

Note that if the nonlinear system is represented as a time varying linear system by allowing τ to vary as $3\bar{x}_1$ the eigenvalues of the linearized model are

$$\lambda_{1,2} = \frac{-1 \pm \sqrt{1 - 4a\tau}}{2} = \frac{-1 \pm \sqrt{1 - 12ay_f^2}}{2} \qquad (B.9)$$

Thus, the matrix $\mathbf{A}$ varies rapidly with time, especially for large commanded inputs. Much of the linear analysis assumes that the plant is considered piece-wise LTI, but it has been shown that this is precisely the type of plant which frequently violates these assumptions [33].

The research in [25] showed that this SISO system can pose quite a problem when dealing with nonlinearities, let alone uncertain parameters. Additional complications and dynamics are induced from combining separate controllers which certainly makes the design task non-trivial. The drastic effects of mis-matched energy between controller and plant, including the proper handling of non-zero initial conditions on either, must be addressed in any attempt to hand-off control during continuous operation [25]. The research in [25] was exploratory in nature and raised such issues. Using the above system some of the concerns for the SISO case are addressed in the sequel. The results of Chapter III answer these concerns and provide a solution that properly addresses the MIMO generalization by means of multiple MISO solutions. Chapter IV then provides the solution for the plant without parameter variation ($a = 1$) with detailed analysis for nonlinear, LTV, and LTI truth models.

B.3   Fuzzy Logic Control of a Family of Two Plants

In this section the application of the Fuzzy Logic concept of blending controllers, or *fuzzy controller scheduling*, is investigated.

The plants under consideration are

$$P_i = \begin{cases} \dot{x}_1 &= -x_1 + x_2 \\ \dot{x}_2 &= -\tau_i x_1 + \tau_i u \qquad i = 1, 2 \\ y &= x_1 \end{cases}$$

where the parameter τ_i is arbitrarily chosen as $\tau_1 = 1$ and $\tau_2 = 2$. Furthermore, suppose that the parameter is dependent on an exogenous "measurable" variable, say v where,

$$\tau = 3v^2$$

Thus, if $v = v_1 = \frac{1}{\sqrt{3}}$ then $\tau = \tau_1 = 1$ and if $v = v_2 = \frac{\sqrt{2}}{\sqrt{3}}$ then $\tau = \tau_2 = 2$. Note that for $v = y$ the output, this is a particular case of the nonlinear problem developed in the last section where the operational envelope (in $\mathbb{R}^2$) has been partitioned into two regions about P_1 and P_2 with fixed $a = 1$. That is, the nonlinear plant without parameter variation is to be addressed. The merits of the FLC approach to scheduling depicted in Figure B.1 is explored, where $G_1(s)$ and $G_2(s)$ are two separate LTI controllers independently designed to control $P_{1,2}(s)$ respectively.

Simulation experiments are performed to "validate" the approach and point out shortcomings which must be addressed by the final synthesis technique.

Three cases are explored in the sequel depending on the fidelity of the plant's truth model. These being an LTI plant, an LTV plant, and the full nonlinear plant.

B.3.1 Linear Time-Invariant Plant.   In the spirit of gain scheduling, first treat the problem as two separate plants; i.e. control P_1 and P_2 and design an independent controller for each. From the model given for a fixed τ, the transfer
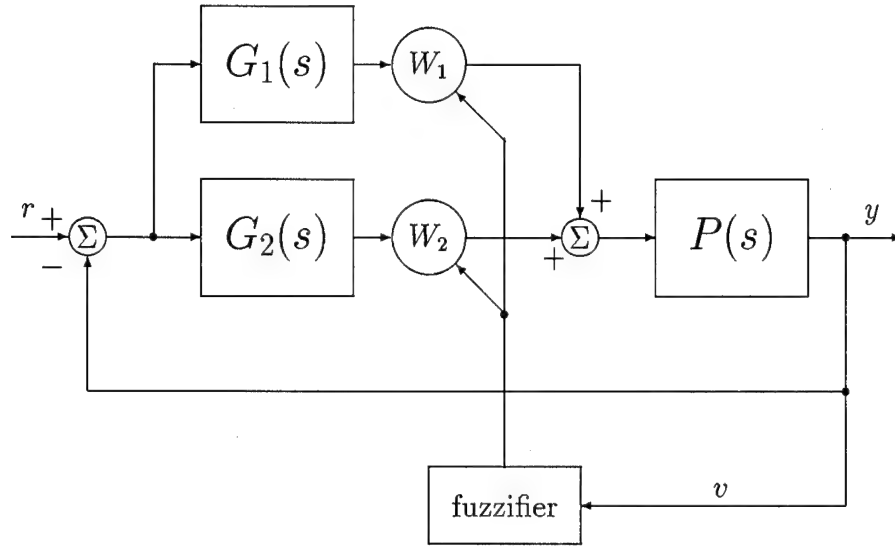
Figure B.1   Adaptive Control Structure for two Point Controller Designs

function of the plant as a function of τ is

$$P_\tau(s) = \frac{\tau}{s^2 + s + \tau} \tag{B.10}$$

A very common and benign appearing system to handle with linear control.

Controllers $G_1(s)$ and $G_2(s)$ are now designed to provide some specified perfor-mance. The chosen desired performance is to track a step input, r, with zero error and have an approximate 2nd order linear response with damping ratio $\zeta = 0.5$, and natural frequency $\omega_n = 2$. That is, the performance specifications for a step input are a 4 second settling time and a 16% overshoot [8]. To design these controllers a technique is used which in general, yields a very poor solution due to inevitable parameter/modeling uncertainty. However, it is shown how *fuzzy controller schedul-ing* can remove such draw backs. The controller is formed by *canceling* the plants
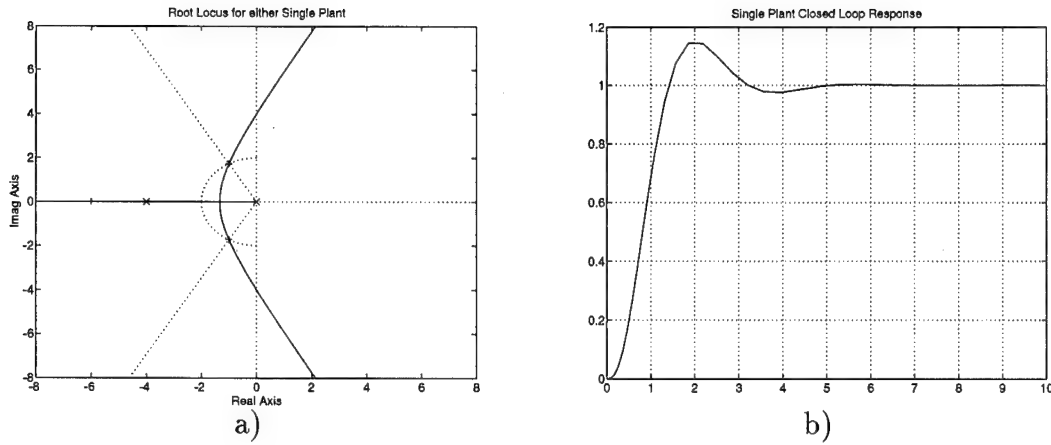
Figure B.2   Root Locus and Closed-Loop Performance for either $P_{1,2}$

dynamics and replacing them with a desired open-loop transfer function.

$$G_i(s) = \frac{\frac{k}{\tau_i}(s^2 + s + \tau_i)}{s(s+4)(s+4)} \qquad (B.11)$$

Root Locus analysis gives $k = 24$ to yield the desired closed-loop performance. The selection of k and the closed-loop response of $G_1(s)P_1(s) = G_2(s)P_2(s)$ are shown in Figure B.2. Note that this form of controller is impractical as $\tau \to 0^+$.

In the remainder of the discussion, reference to the run numbers listed in Table B.1 is made. These points are obtained by first linearly varying v over it's range between the two plants, and then linearly varying τ over it's range. So the nominal cases, those that yield the response of Figure B.2b, are run #7 for P_1 and run #8 for P_2.

As noted earlier, these designs in a point-wise or classical gain scheduling context are really mathematical trickery and can not normally be considered as feasible since they rely on perfect cancellation of the plants' dynamics. It is well known that this is not really possible since small deviations in τ from the designed

B-9

Run #	v	τ
1	0.6371	1.218
2	0.6969	1.457
3	0.7567	1.718
4	0.6455	1.25
5	0.7071	1.5
6	0.7637	1.75
7	0.5774	1
8	0.8165	2

Table B.1   Run Number vs. τ and v $(\tau = 3v^2)$

cases leave dominant poles near the original poles. That is, for any of the runs listed, except for the nominal cases, one can expect performance degradation. This is shown in Figure B.3 where a single compensator is used to control $P(s)$ and τ varies (for both G_1 and G_2) as in Table B.1.

It is seen that either design by itself is unacceptable. However, when posed in the setting of the problem statement, the results are entirely different. The *fuzzy scheduler* is implemented in SIMULINK® as in Figure B.4. The state space block of the figure is constant during the entire simulation using either τ_1 or τ_2 depending on one's objective. The fuzzy logic inference used is merely the fuzzification of the crisp measurement of v, since only weighing the amount of control effort from each controller to apply is used. That is, the blending of the controllers is performed in Figure B.4 by giving each point design an amount of control authority based upon the current operating point's degree of membership in the fuzzy variable $\mathcal{P}_i$. Placing this in the standard rule statement one has:

IF *Plant* is $\mathcal{P}_i$, **THEN** $W_i = 1$ for $i = 1, 2$.

Rule conflicts are settled using a non-normalized summation of the two weighted controller outputs.

The scheduling is performed using both the triangular and Gaussian membership functions $(\mu_1(v), \mu_2(v))$ shown in Figure B.5. The functions are fit such that

B-10

a) Output of P_1 using G_2

b) Output of P_2 using G_1

c) Varying Plant using G_1

d) Varying Plant using G_2

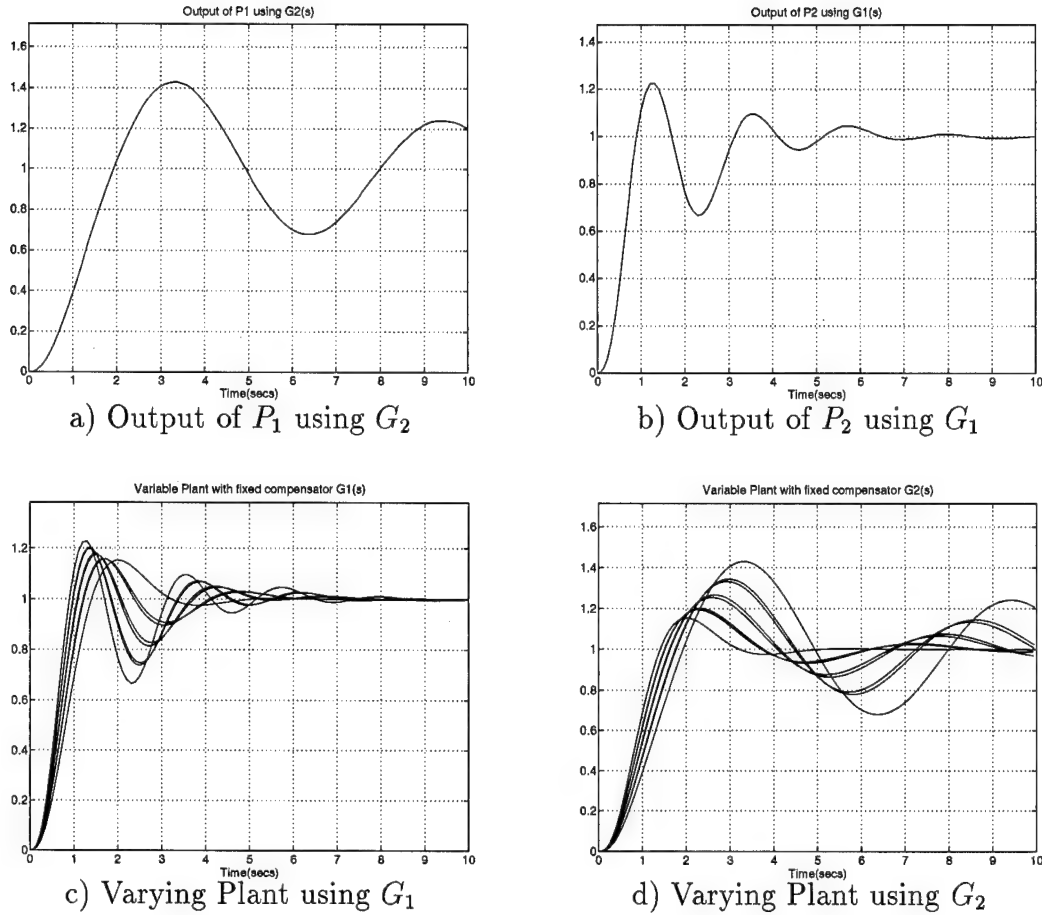Figure B.3   System Response using Single Fixed Controller Over Range of τ

both controllers are equally weighted at the midpoint of v's universe of discourse, $v_v = \frac{1}{2}(\sqrt{2/3} - \sqrt{1/3}) \approx 0.6969$. This is dictated by use of triangular membership functions (MFs) if total overlap of the MFs is desired, and allows the same variance for both MFs to be used in the Gaussian case. However, the controllers are designed in τ-space and it "seems more correct" to have $\mu_1(v_\tau) = \mu_2(v_\tau) = 0.5$, where $v_\tau = \sqrt{\frac{1.5}{3}} \approx 0.7071$. For the examples to follow, only the centering at v_v is used. Optimal selection of this *trade off* point must be addressed in the final design technique.
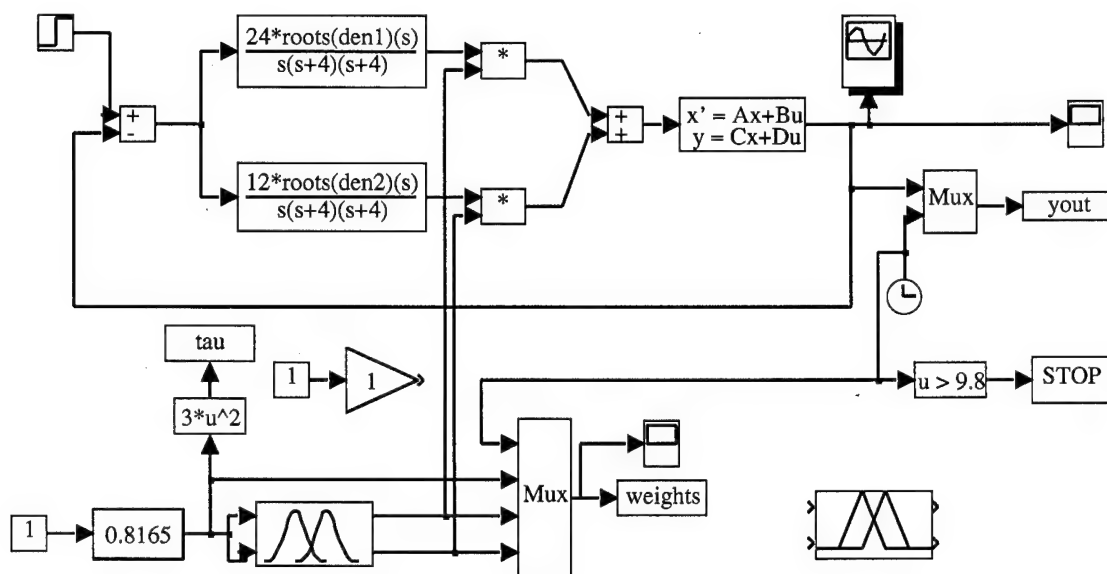
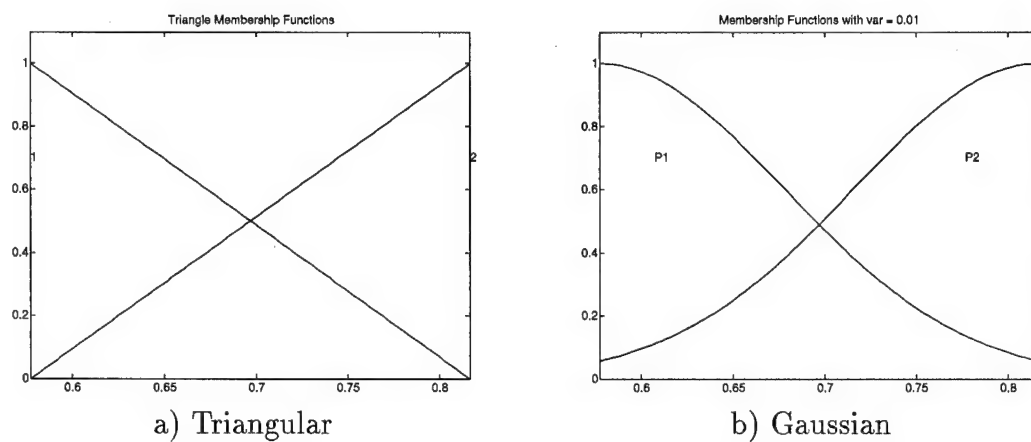Figure B.4   Block Diagram of Fuzzy Scheduler

a) Triangular

b) Gaussian

Figure B.5   Membership Functions used for v
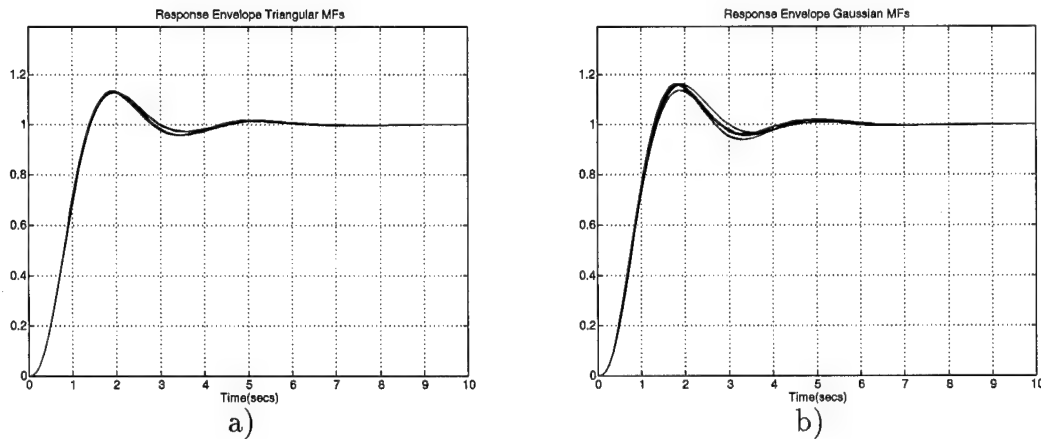
B-12

Figure B.6   Fuzzy Scheduled Controller Response Over Applicable Range of τ

The results using both MFs as τ varies according to Table B.1 are excellent and are shown in Figure B.6. The resulting fuzzy scheduler displays nearly invariant response over the entire range of τ despite the nonlinear relationship between v and τ. A comparison using MFs based upon v_τ is not performed. It is noted that the triangular MF based controller out performs the Gaussian one in the sense that less variation in output is observed. It is felt that this is due to the inherent normalization of the weights across the universe of discourse. That is, since $W_i = \mu_i(v)$, for the triangular MFs, $W_1 + W_2 = 1$. Although the weights for the Gaussian MFs can be normalized, no attempt is currently made since it's response is so good. The effects of normalization increase as the dimension of spaces increase due to the use of a "product rule" in multivariate FL, as developed in Chapter II, and the fact that any individual membership is bounded above by unity.

In the analysis so far, only the uncertainty due to the variation in τ which is generated by the underlying nonlinearities is addressed. A controller must also deal with model parameter uncertainty. Experimentation is made to explore the effects of the parameter a from Eq. (B.6) on page B-5. Since no attempt is made to design
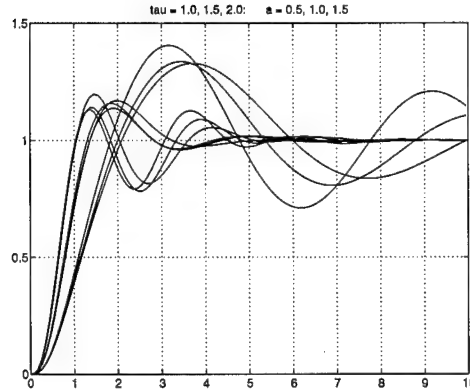
Figure B.7 Thumb Print Response of System for $\tau \in \{1, 1.5, 2\}$, $a \in \{0.5, 1.5\}$

for a, it is no surprise that its variation drastically effects the system's response as shown in the "thumb print" in Figure B.7.

However, one can gain insight from these simulations through Figure B.8. As previously demonstrated, the design is very robust for variations in $\tau \in [1, 2]$. Now for the three fixed $\tau, \in \{1, 1.5, 2\}$ vary the parameter $a \in \{0.5, 1, 1.5\}$. The results are shown in Figures B.8a-c, and are not impressive. If instead one views the responses by collecting on similar values for a the results in Figures B.8d-f are obtained. These figures show the common dynamics as a function of a. They suggest that if the uncertainty in a were treated as that of τ, one should be able to schedule on a to obtain the desired response over the uncertain region in this 2-dimensional parameter space. Thus, this uncertain SISO plant becomes a MISO fuzzy scheduling problem.

B.3.2 Linear Time-Varying Plant.   The next level of increased model fidelity is to implement Eqs. (B.6)-(B.8) on page B-5 with a dynamic τ, representing incremental changes in the trim point. This yields an LTV plant. Now, define $\tau(t) = 3x_1^2(t)$ where the perturbation state $x_1^2 = y$ varies as the output during the simulations. In the sequel, the notation indicating τ's dependence on time is dropped
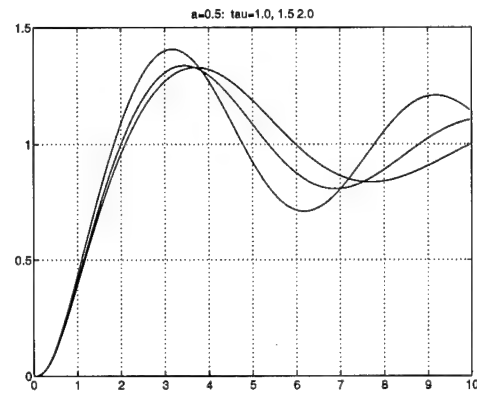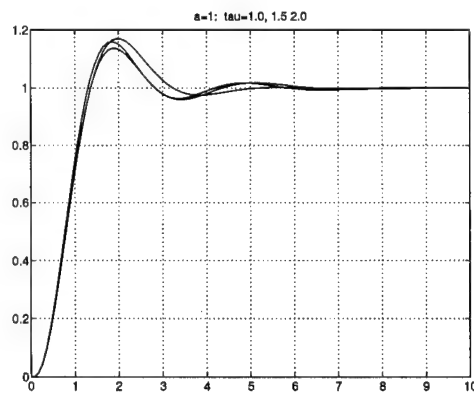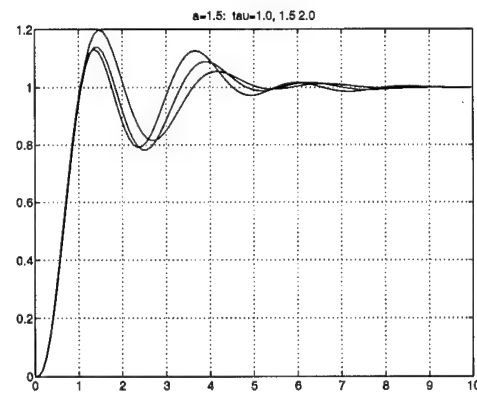
B-14
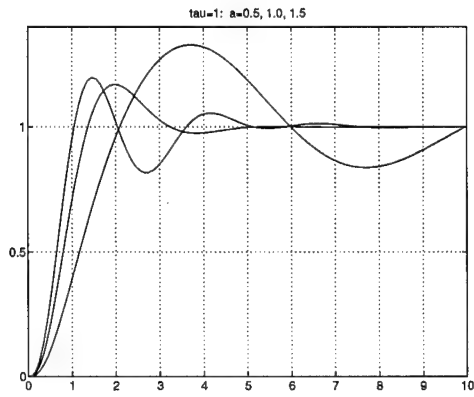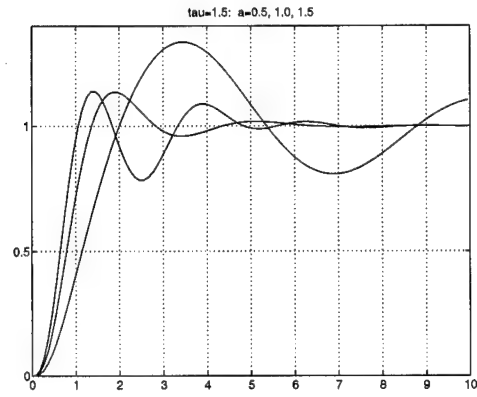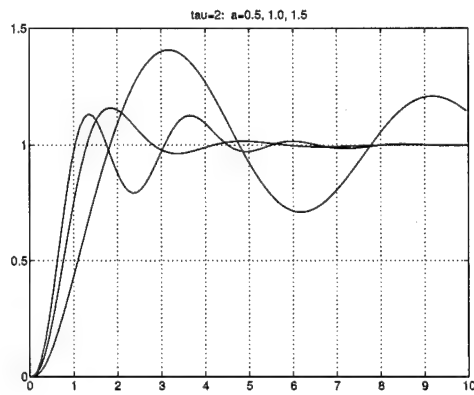
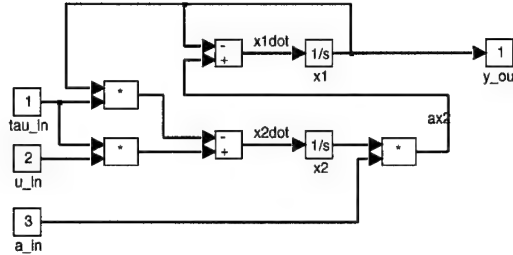Figure B.8   Clustering of Response for $\tau \in \{1, 1.5, 2\}, a \in \{0.5, 1.5\}$

Figure B.9   LTV Plant with Externally Defined τ

for brevity. Where as Eq. (B.7) is linear in τ, the LTV plant is nonlinear in x and u. The SIMULINK® block diagram for the LTV plant with externally defined τ is shown in Figure B.9.

The current two point controller only applies "sufficient cover" of the parameter space on the interval $[\tau_1, \tau_2]$, therefore simulations should be restricted to this range to have any merit. Clearly this is of no concern for the LTI truth model. Therefore, the simulations are performed starting from the trim point represented by τ_1, with step inputs up to the trim point representing τ_2. Since τ already varies as the output, the indication of acceptable performance is the response for an admissible command input. The simulation diagram is depicted in Figure B.10. To avoid solving for the initial conditions on all of the compensator states, a nominal input of $\bar{u} = \sqrt{1/3}$ is applied to the LTV plant (Uo in the simulation diagram). The plant's initial conditions are set at, referring to the linearization, $\bar{x}_1 = a\bar{x}_2 = \bar{u}$, with $a = 1$.

The response to 4 admissible commanded step inputs is shown in Figure B.11. The system shows good, approximately linear, responses for the smaller three commands. Slight degradation is noted on the largest command which, although it may still be judged acceptable, has a final value of $\sqrt{2/3}$ representing τ_2. A reason for
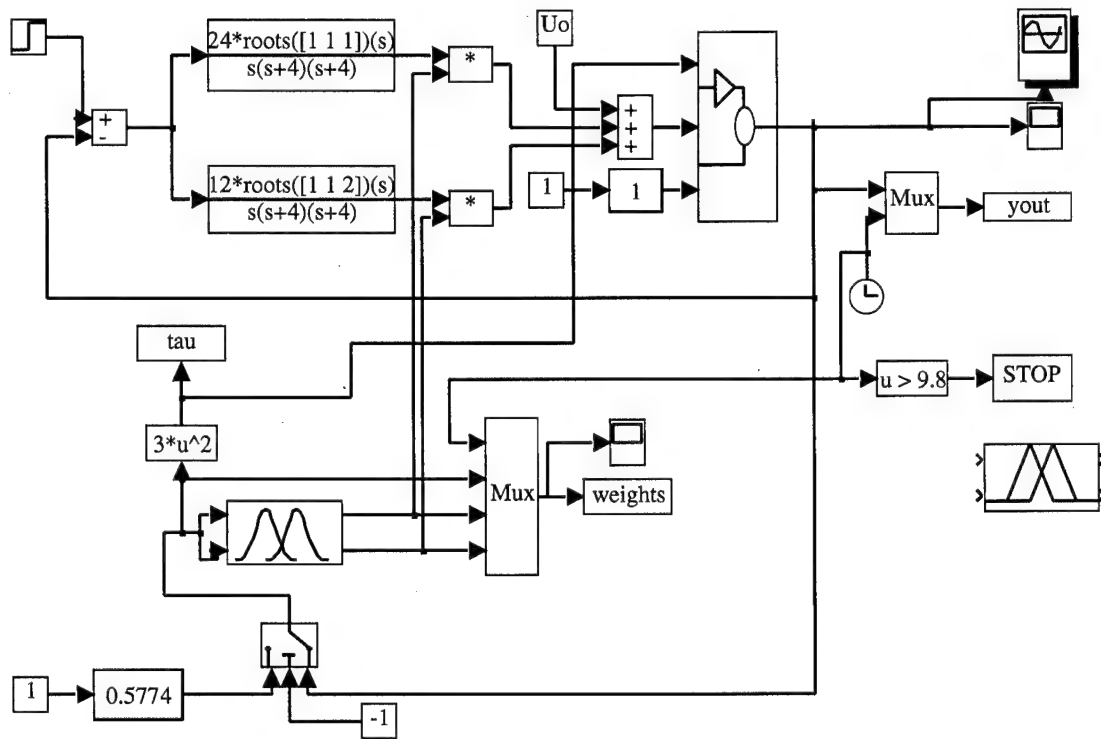
B-16

Figure B.10   Simulation Diagram for LTV Plant

the change in performance may be that this command forces $\tau > \tau_2$ and hence out of the designed for region in P. Clearly the LTI model is inadequate to describe this system, but its inclusion in analysis stages provides a larger set of mathematical tools to gain insight.

B.3.3   Nonlinear Plant.   Finally the fuzzy scheduler's ability to control the "true" nonlinear plant is investigated. The plant is given by Eqs. (B.3)-(B.5) on page B-4 and is implemented in SIMULINK® as in Figure B.12. For reasons given earlier, the simulations are restricted to the same interval as for the LTV case, $[\tau_1, \tau_2]$, and the same commanded inputs are used. The simulation is implemented as in Figure B.13 and the system response is shown in Figure B.14.

B-17
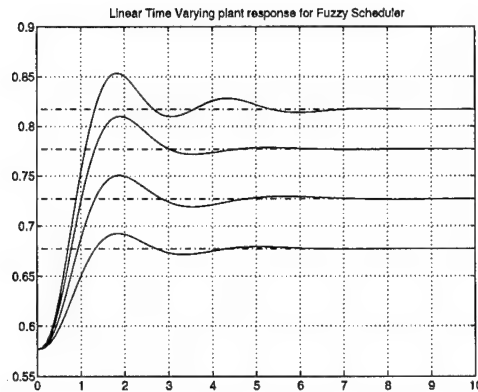
Figure B.11   LTV Plant Response of Fuzzy Scheduler

Note: Check initail conditions on integrators function of Uo and astart
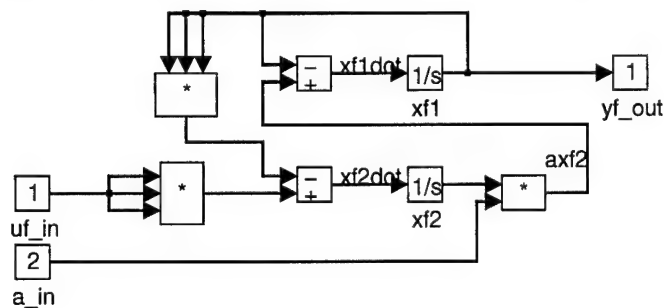
Figure B.12   Nonlinear Plant Implementation

Clearly the response is unacceptable, and the strong nonlinearity of the plant is evident. Design techniques that make performance claims based upon only the LTI models do not hold in the real world for this plant. Although the LTI models are "good enough" in sufficiently small regions about the nominal design points, these regions *must* be quantified, and the proper means is by way of nonlinear simulation.

Figure B.13   Simulation Diagram for Nonlinear Plant

Figure B.14   Nonlinear Plant Response of Fuzzy Scheduler

B-19

B.4 Summary

The approach for this system shows promise of interesting and fruitful results for the LTI and LTV truth models. The Linear (in τ) Time-Varying case does warrant closer investigation given the above performance. The nonlinear system's performance visualizes the difficulty facing the proposed technique a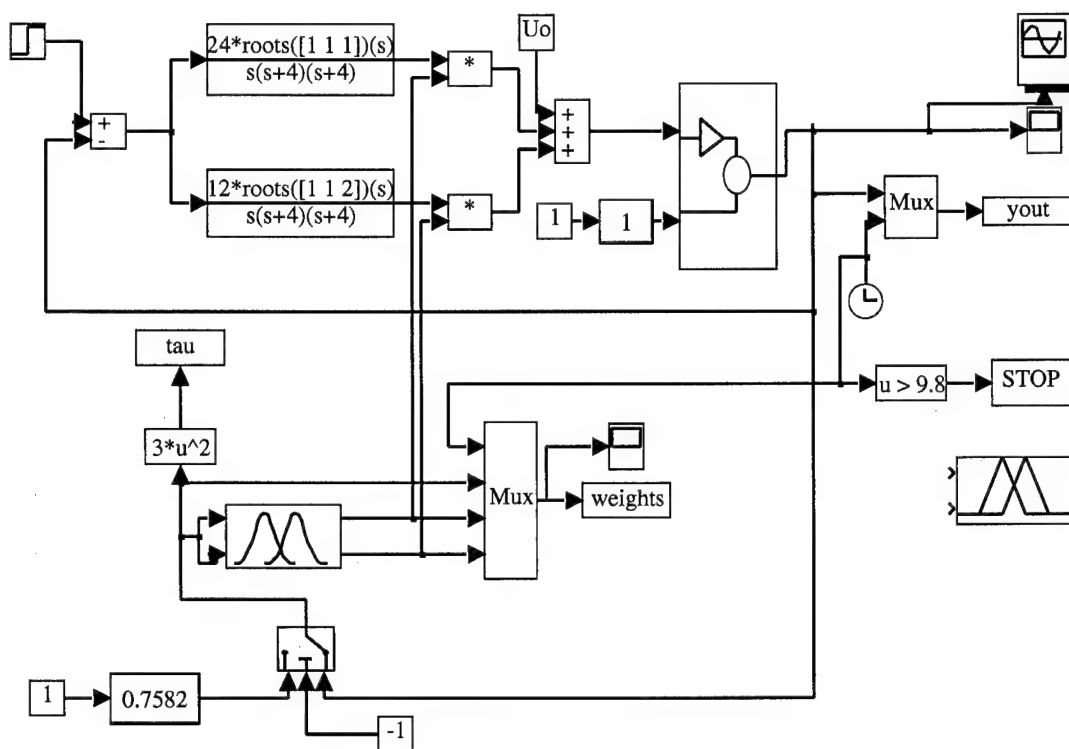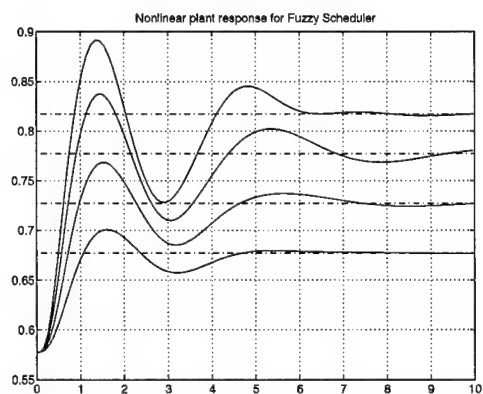nd demonstrates the requirement of validation by simulation. Rigorous mathematical analysis may provide promises of stability or even asymptotic tracking, both of which the current design exhibits. However, transient response dictates the acceptability of the design and the simulations quantify the current designs deficiencies. There are also additional, more subtle, points of interest.

There are additional complexities that will be inherent in real world problems that are not addressed in this example. For instance plant dynamics being a function of more than one variable that "requires" scheduling. How does one combine or weight the individual controllers then? In most techniques complexity increases as the model order increases to a MIMO controller. This is not the case here, *assuming that the fixed MIMO controllers have been built*, the complexity increases with the number of scheduling measurements (dimension of parameter space P) and how does one combine these to assign weights to the individual controllers. These concerns are properly addressed by the solution given in Chapter III.

Also the small universe of discourse did not excite the nonlinearity in τ very much. Of course in theory one should be able to keep this under control by sufficiently partitioning of the operating envelope. This may lead to a lot of point designs and thus more complexity. However, this is not really any different then any problem that "requires" scheduling. As long as the concerns of the previous paragraph can be addressed, the only added complexity is the fuzzification of measurements and there are commercial chips which handle this in a parallel fashion [39, 46].

Of course there are the standard difficulties of any nonlinear design technique such as stability. Short of a totally analytic solution, these concerns are best addressed by extensive, realistic simulation.

Appendix C. Support Data for Chapter III

C.1 Listing of MATLAB Function voronoi.m

```
function [vertices,numvert]=voronoi(D,A,domain,p)
% voronoi  Find the voronoi diagram of D=(x,y;), a set of at least 3 points
%   in R^2. Each row of D must be a unique 2-D vector.  A is the square
%   Adjacency matrix representing the delaunay diagram where A(i,j)=j if node
%   i, D(i,:)=(x(i),y(i)), is connected to node j and A(i,j)=0 otherwise. If
%   A is empty it is generated using A=delaunay(D). domain is the region of
%   interest in R^2 specified as domain=[min_x max_x min_y max_y].  If omitted
%   one based on D is used (see below for details).  If no output arguments
%   are specified or p=1, a plot generated.
%   CALLS con_hull.m and optionally delaunay.m
%
%   Notes on format of output.  Currently the 2 outputs are: 1) numvert, an
%   n element column vector (n= # of distinct points) of index  information.
%   Where numvert(i) is the number of vertices of the Voronoi cell for point
%   i specified by D(i,:). 2) vertices, a sum(numvert) by 2 matrix containing
%   Voronoi cell vertices appended in order of the elements of D. So the
%   vertices of the cell for point 1 are vertices([1:numvert(1)],:), and for
%   the i^th cell, i = 2,...,n the vertices are
%   vertices([sum(numvert([1:i-1]))+1:sum(numvert([1:i]))],:)
%   For cells that lie on the convex hull, the first and last vertex are used
%   to identify the unbounded polygon forming the cell.  Their output depends
%   upon the input domain specified. When domain is specified, the cells
%   are vaild only in the region of R^2 specified by domain.  If domain is
%   unspecified (empty), infinit bisecting rays are indicated in yellow in the
%   plot and the following holds. If the ray forming a side of the cell is
%   horizontal or vertical, the vertex is correct for any element of R^2;
%   otherwise it is merely directionally correct (+/-inf).  This is useful to
%   provide a flag to such cells and simplify closing of the cells in the plot.
%   Either way can give suspect results when a vertex of an interior cell
%   exceeds the domain (specified or default which is the axis limits of plot).
%   Increasing the axis beyond the domain after running voronoi can also be
%   confusing. The work around is to specify a large enough domain, rerun,
%   than expand the plot by changing the axis.
%
%   This isn't really the best way to deplict the information, since it has 2
%   main problems. 1) We have to calulate a single finite vertex at least 3
%   seperate times (infinite twice). 2) Because of this, finite precision
%   may result in different answers for the same index.  Both of these could
%   be solved by a different data structure.  In particular, construct an
%   adjacency matrix and make a queue of vertex labels.  Then as each vertex
%   is solved remove its label from the queue.  Now each vertex is solved for
%   once and obiviously unique.  However, since I use the Delaunay Diagram
%   for all my calculations I haven't bothered solving the adjacency problem.
%
% see refs for developement and proof of completness
% 1) Du, Ding-Zu and Hwanf, F., Computing in Euclidean Geometry,
%    World Scientific, 1992, QA447.C573 p. 210
% 2) Guibas & Stolfi, ACM Tans. on Graphics 4(2):74-123, 1985
%
%    [vertices,numvert]=voronoi(D,A,domain,p)
%
% NAME:  voronoi
% LAST REVISION: 21 Nov 94  MatLab 4.2
% Author: TOM KOBYLARZ  Air Force Institute of Tech WPAFB, OH
%         tkobylar@afit.af.mil

% figure out input format and output requested by parameters provided
if nargout==0,pl=1;else,pl=0;end
```

```
if (nargin >4),eval(['help voronoi']), error('wrong number of input arguments.'); end
if nargin==0,eval(['help voronoi']),return; end
npts=length(D);
if nargin==1,A=[];domain=[];end
if nargin==2,domain=[];end
if nargin==4,p1=p;end
if size(D,2)~=2
  eval(['help voronoi']);
  error('     D must contain 2 columns, each row a 2-D vector');
end
if npts < 3
  eval(['help voronoi']);
  error('     D must contain at least 3 sets of points (x,y)');
end
if isempty(A)      % If delaunay diagram of D not given create it
  A=delaunay(D);
else
  if (npts ~= size(A,1)) | (npts ~= size(A,2))
    eval(['help voronoi']);
    error('     A must be square and of the form returned by A=delaunay(D)');
  end
end

if isempty(domain)
  domainflag=0;
else
  domainflag=1;
  if (domain(1)>=domain(2)) | (domain(3)>=domain(4)) | (length(domain)~=4)
    eval(['help voronoi']);
    error('     domain must be of form domain=[min_x max_x min_y max_y]');
  end
end

hull=con_hull(D,'i'); % get hull to catch unbounded regions of hull points
vertices=[];
numvert=zeros(npts,1);

% dom, ax and infrays are used to display infinite rays from hull point
% cells. See help for technique to use these to obtain finite but large
% vertices. dom is the domain of interest, infrays are line segments of
% the infinite rays, and ax is the axis used in the plot
ax=[min(D(:,1)) max(D(:,1)) min(D(:,2)) max(D(:,2))];
ax=[ax(1)-abs(.1*(ax(2)-ax(1))) ax(2)+abs(.1*(ax(2)-ax(1))) ax(3)-abs(.1*(ax(4)-ax(3))) ax(4)+abs(.1*(ax(4)-ax(3)))];
infrays=[];
if domainflag==1
  dom=domain;
else
  dom=ax;
end

% Generate the seperating hyperplanes for each unique point. By shifting
% to the origin we can find bisecting midpoints and perp slope easily

for i=1:npts
  neigh=find(A(i,:)); % only interested in nearest neighbors from A
  Dn=D(neigh,:);
  z=.5*[Dn(:,1)-D(i,1) Dn(:,2)-D(i,2)]; % the bisector mid point
  % place perp slope in third column, for no duplicates (0,0) not in z
  z(:,3)=-1*z(:,1)./z(:,2);
  % obtain a cyclic ordering to find intersections
  rays=atan2(z(:,2),z(:,1));        % atan2 returns +/- pi
  wrap=find(rays<0);                % wrap correction to keep 0-2pi
  rays(wrap)=rays(wrap) + 2*pi * ones(size(wrap),1);
  [junk,I]=sort(rays);
  neigh=neigh(I);                   % redorder all working variables
```

```
z=z(I,:);
ishull=(find(i==hull));
if ~isempty(ishull)
  % do stuff here to avoid solving for intersection of 2 hull bisectors
  if ishull==length(hull)
    startat=hull(1);
  else
    startat=hull(ishull+1);
  end
  first=find(neigh==startat);
  if first~=1 % if it is no need to reorder
    I=[first:length(neigh) 1:first-1];
    neigh=neigh(I);  % reorder all working variables
    z=z(I,:);
  end
  % Now for a representation of the vertices at infinity
  infvert=zeros(2,2);
  hn=[1 length(neigh)];
  ray=atan2(z(hn,2),z(hn,1));
  if ray(2)>0 % swap orientation since reference from current pt
    ray(2)=ray(2)-pi;
  else
    ray(2)=ray(2)+pi;
  end
  for k=1:2
    if abs(ray(k))<eps
      infvert(k,:)=[z(hn(k),1)+D(i,1) -inf];
    elseif abs(abs(ray(k))-pi)<eps
      infvert(k,:)=[z(hn(k),1)+D(i,1) inf];
    elseif abs(ray(k)-pi/2)<eps
      infvert(k,:)=[inf z(hn(k),2)+D(i,2)];
    elseif abs(ray(k)+pi/2)<eps
      infvert(k,:)=[-inf z(hn(k),2)+D(i,2)];
    elseif ray(k)>0 & ray(k)<pi/2
      infvert(k,:)=[inf -inf];
    elseif ray(k)>pi/2 & ray(k)<pi
      infvert(k,:)=[inf inf];
    elseif ray(k)>-pi & ray(k)<-pi/2
      infvert(k,:)=[-inf inf];
    elseif ray(k)>-pi/2 & ray(k)<0
      infvert(k,:)=[-inf -inf];
    end
  end
else     % we will be wrapping around interior points
  neigh=[neigh neigh(1)];
  z=[z;z(1,:)];
end
% solve for intersections of perp bisectors
vertex=zeros(length(neigh)-1,2);
for k=1:length(neigh)-1
  bi1=z(k,[1,2])+D(i,:);
  bi2=z(k+1,[1,2])+D(i,:);
  if finite(z(k,3)) & finite(z(k+1,3))
    x=(z(k,3)*bi1(1)-bi1(2)-z(k+1,3)*bi2(1)+bi2(2) )/(z(k,3)-z(k+1,3));
    y=z(k,3)*(x-bi1(1)) + bi1(2);
  else                    % catch vericle bisectors assuming no duplicates
    if finite(z(k,3))     % k+1 must be verticle
      x=bi2(1);
      y=z(k,3)*(x-bi1(1)) + bi1(2);
    else
      x=bi1(1);
      y=z(k+1,3)*(x-bi2(1)) + bi2(2);
    end
  end
  vertex(k,:)=[x y];
```

C-3

```
      end % next k
      if ~isempty(ishull)  % add representation of infinite vertices for hull pts
        bi=z(hn,[1,2])+[D(i,:);D(i,:)];
        x=dom([infvert(:,1)>0]+[1;1])';  % select the correct directions
        y=z(hn,3).*(x-bi(:,1)) + bi(:,2);
        if isinf(y(1))                   % catch verticle ccw rays
          y(1)=dom(sign(sign(infvert(1,2))+1)+3);
          x(1)=vertex(1,1);
        end
        if isinf(y(2))                   % catch verticle cw rays
          y(2)=dom(sign(sign(infvert(2,2))+1)+3);
          x(2)=vertex(length(neigh)-1,1);
        end
        infrays=[infrays; vertex(1,:) x(1) y(1)];
        if domainflag==1
          infvert=[x y];
        end
        vertex=[infvert(1,:);vertex;infvert(2,:)];
      end
      numvert(i)=length(vertex);
      vertices=[vertices;vertex];
    end % next i

    if pl==1
      clf
      start=1;
      for i=1:npts
        stop=start+numvert(i)-1;
        vertex=vertices([start:stop],:);
        plot([vertex(:,1);vertex(1,1)],[vertex(:,2);vertex(1,2)],'r')
        hold on
        start=stop+1;
      end
      for N=1:npts,text(D(N,1),D(N,2),int2str(N)),end
      axis(ax)
      if domainflag~=1 % don't plot redundant data
        for i=1:length(hull) % plot the inf rays from hull edges
          plot([infrays(i,1),infrays(i,3)],[infrays(i,2),infrays(i,4)])
        end
      end
      hold off
      title('Voronoi Diagram')
    end
```

C.2 Listing of MATLAB Function delaunay.m

```
function A=delaunay(D,p)
%delaunay  Find the delaunay diagram of D=(x,y), a set of at least 3 points
%    in R^2. Each row of D must be a unique 2-D vector. If no output arguments
%    specified or p=1, a plot generated.  The returned A is the Adjacency matrix
%    representing the diagram where A(i,j)=j if node i, D(i,:)=(x(i),y(i)), is
%    connected to node j and A(i,j)=0 otherwise. Use gplot(A,D) to obtain plot.
%    CALLS triangle.m, con_hull.m, ccw.m, incircle.m
%
% see refs for developement and proof of completness
% 1) Du, Ding-Zu and Hwanf, F., Computing in Euclidean Geometry,
%     World Scientific, 1992, QA447.C573 p. 210
% 2) Guibas & Stolfi, ACM Tans. on Graphics 4(2):74-123, 1985
%
%     A = delaunay(D,p)
%
%  NAME:  delaunay
%  LAST REVISION: 13 Dec 94  MatLab 4.2
%  Author: TOM KOBYLARZ  Air Force Institute of Tech WPAFB, OH
%           tkobylar@afit.af.mil


% figure out input format and output requested by parameters provided
if nargout==0,pl=1;else,pl=0;end
if (nargin >2),eval(['help delaunay']), error('wrong number of input arguments.'); end
if nargin==0,eval(['help delaunay']),return; end
if nargin==2,pl=p;end
if length(D)<3
  eval(['help delaunay']);
  error('     D must contain at least 3 sets of points (x,y)');
end
if size(D,2)~=2  % try to get D in required format
  D=D';
end
if size(D,2)~=2  % check to see if ok now
  eval(['help delaunay']);
  error('     D must contain 2 columns, each row a 2-D vector');
end


[tri,hull]=triangle(D); %Get Indices of cw traingles and ccw hull pts.
if isempty(tri)        % all points were colinear manually build A
  disp(' !! All points are colinear !!')
  npts=length(hull);
  A=zeros(npts);
  A(hull(1),hull(2))=hull(2);                 % ends pts only have 1 neighbor
  A(hull(npts),hull(npts-1))=hull(npts-1);
  for i=2:npts-1                              % all others have 2
    A(hull(i),hull(i-1))=hull(i-1);
    A(hull(i),hull(i+1))=hull(i+1);
  end
  return
end
Dtri=[tri(:,1) tri(:,3) tri(:,2)]; % convert to ccw triangles
nt=size(Dtri,1);

% Now form a queue of all non-directional interior edges of Dtri
edges=[Dtri(:,[1 2]); Dtri(:,[2 3]); Dtri(:,[3 1])];
temp=sort(edges')'; % make non-directional i.e. (1 2)=(2 1)
% sort by x when a tie occurs sort on y. So sort on y first
[junk,yi]=sort(temp(:,2));
[junk,I]=sort(temp(yi,1));
xi=yi(I);
edges=temp(xi,:); % The ordered redundant edges
% remove duplicates
```

```
dups=find(diff(edges(:,1))==0 & diff(edges(:,2))==0);
edges(dups+1,:)=[];

%Test to ensure we have all edges
n=length(D);            % number of points (assuming no duplicates)
k=length(hull);         % # points on final hull
t=3*(n-1)-k;            % # of expected triangles
if t~=size(edges,1)
  disp([' ']);disp('  Number of edges is incorrect, suspect duplicate points');
end


% now form the hull edges and remove from queue
hulle=[hull [hull([2:length(hull)]);hull(1)]];
hulle=sort(hulle')';  % make non-directional i.e. (1 2)=(2 1)
hei=[];
for i=1:size(hulle,1)
  hei=[hei;find(edges(:,1)==hulle(i,1) & edges(:,2)==hulle(i,2))];
end
edges(hei,:)=[]; % the interior non-directional edges

degen=[];          % initialize as no degenerative (co-circular) pts

% use flip routine while edges is not empty
while ~isempty(edges)
  edge=edges(1,:);
  del_edge=find(edges(:,1)==edge(1) & edges(:,2)==edge(2)); % find all
  edges(del_edge,:)=[]; % remove occurances although it may be added later
  % get the triangles that share edge
  opp=find((edge(1)==Dtri(:,1)|edge(1)==Dtri(:,2)|edge(1)==Dtri(:,3)) &...
           (edge(2)==Dtri(:,1)|edge(2)==Dtri(:,2)|edge(2)==Dtri(:,3)));
  if length(opp)~=2,error(' Exactly 2 triangles share an interior edge');end
  tri1=Dtri(opp(1),:);
  tri2=Dtri(opp(2),:);
  c1=find(tri1~=edge(1) & tri1~=edge(2));
  c2=find(tri2~=edge(1) & tri2~=edge(2));
  % Now check if ABCD locally Delaunay as is
  test=incircle(D(tri1(1),:),D(tri1(2),:),D(tri1(3),:),D(tri2(c2),:));
  if test==1
    new1=[edge(1) tri1(c1) tri2(c2)]; % new tri vertices (make ccw)
    if ccw(D(new1(1),:),D(new1(2),:),D(new1(3),:))~=1,new1=new1([2,1,3]);end
    new2=[tri1(c1) tri2(c2) edge(2)];
    if ccw(D(new2(1),:),D(new2(2),:),D(new2(3),:))~=1,new2=new2([2,1,3]);end
    Dtri(opp(1),:)=new1; % update the triangulirization
    Dtri(opp(2),:)=new2;
    % Now add supect edges of quad involved in flip to queue excluding those on the hull
    qhull=[tri1 tri2(c2)];
    temp=con_hull(D([tri1 tri2(c2)],:),'i');
    qhull=qhull(temp);
    temp=[qhull; [qhull([2:4]) qhull(1)]];
    qhull=sort(temp')';  % make non-directional i.e. (1 2)=(2 1)
    hei=[];
    for i=1:4
      if ~isempty(find(hulle(:,1)==qhull(i,1) & hulle(:,2)==qhull(i,2))),hei=[hei;i];end
    end
    qhull(hei,:)=[]; % the interior non-directional edges
    edges=[edges;qhull]; % add suspect to queue
    % disp(['  Swapped diagonal and added ',int2str(size(qhull,1)),' suspect edges to the queue'])
  elseif test== -1
    % disp('  locally Delaunay')
  else
    junk=[tri1(1),tri1(2),tri1(3),tri2(c2)];
    disp(['  points [',sprintf('%3.2g',junk),' ] are cocircular'])
    degen=[degen;edge];
  end
end  % all edges deleted and we are done
```

```
% currently we have a Delaunay Triangularization which is not the
% Delaunay Diagram when degenerative cases exist. First we'll
% generate the Adjacency Matrix A to Represent the Triangularization
% where A(i,j)~=0 iff node i is connected to node j. I set A(i,j)=j.

nn=length(D); % assumes no duplicate nodes in D
A=zeros(nn,nn);
for i=1:nn
  junk=find(i==Dtri(:,1)|i==Dtri(:,2)|i==Dtri(:,3));
  temp=Dtri(junk,:);
  junk=temp(:);
  temp=sort(junk);
  delnode=find(diff([0;temp])==0 | temp==i);
  temp(delnode)=[];
  A(i,temp)=temp';
end

% Now remove any extra edges (stored in degen) from degenerate cases
% to form the Diagram by zeroing them out in A
for i=1:size(degen,1)
  A(degen(i,1),degen(i,2))=0;
  A(degen(i,2),degen(i,1))=0;
end

if pl==1
  gplot(A,D)
  hold on
  for N=1:size(D,1),text(D(N,1),D(N,2),int2str(N)),end
  hold off
  title('Delaunay Diagram')
end
```

C.3 Listing of MATLAB Function `triangle.m`

```
function [Dtri,hull]=triangle(D,p)
%triangle  Find a triangularization of D=(x,y), a set of at least 3
%    non-colinear points in R^2. Each row of D must be a 2-D vector. If no
%    output arguments are specified or p=1, a plot generated.  The optional
%    returned vector hull contains the ccw indices returned from
%    con_hull(D,'i').  Dtri is a tx3 matrix where
%     t = 2*((# of nonredundant pts in D)-1)-(# of hull points).
%    Each row of Dtri contains indices of D which form a cw triangle. A ccw
%    set of triangles can be obtained by switching 2 adjacent columns of Dtri.
%
%    [Dtri,hull]=triangle(D,p)
%
%  NAME:   triangle
%  LAST REVISION: 12 Jan 94  MatLab 4.2
%  Author: TOM KOBYLARZ  Air Force Institute of Tech WPAFB, OH
%          tkobylar@afit,af.mil

%  figure out input format and output requested by parameters provided
if nargout==0,pl=1;else,pl=0;end
if (nargin >2),eval(['help triangle']), error('wrong number of input arguments.'); end
if nargin==0,eval(['help triangle']),return; end
if nargin==2,pl=p;end
if length(D)<3
  eval(['help triangle']);
  error('      D must contain at least 3 sets of points (x,y)');
end
if size(D,2)~=2  % try to get D in required format
  D=D';
end
if size(D,2)~=2  % check to see if ok now
  eval(['help triangle']);
  error('      D must contain 2 columns, each row a 2-D vector');
end

% sort by x when a tie occurs sort on y. So sort on y first
[oD,xi]=sort_nd(D,[],100);
dups=find(abs(diff(oD(:,1)))<=100*eps & abs(diff(oD(:,2)))<=100*eps);
if ~isempty(dups)
  disp(['      Duplicate Points Removed']);
  oD(dups+1,:)=[];
  xi(dups+1)=[];
end

% build up triangularization by adding sites. The vertices are s_{i},
% s_{i-1}, and another on the bounadry facing s_{i}. I will generate cw
% triangles below so start with same orientation as rest will be made

% requires special handling if the first 3 or more sorted points are colinear
flag=0;   % colinear flag
tri=[];
j=2;
while flag==0
  ftri=ccw(oD(1,:),oD(j,:),oD(j+1,:));
  if ftri== 1 % 1st 3 are ccw
    tri=[tri;j+1 j j-1];
    flag=1;
  elseif ftri== -1 % 1st 3 are cw
    tri=[tri;j-1 j j+1];
    flag=1;
  else
    tri=[tri;j-1 j 0]; % these 3 are colinear and no triangle formed yet
  end
  j=j+1;     % shift another point
```

```
    if j+1 > length(oD) & flag==0
      hull=con_hull(D,'i');
      Dtri=[];
      disp(' !!  All points are colinear: no triangulization, only hull returned  !!');
      return
    end
  end
oldhull=[1:j]'; % intialize first non-colinear hull
% now replace zeros in tri with first non-colinear point
junk=find(tri==0);
if ~isempty(junk)  % have to make sure the first couple are cw
  tri(junk)=j+zeros(1,length(junk));
  for i=1:size(tri,1)-1
    if ccw(oD(tri(i,1),:),oD(tri(i,2),:),oD(tri(i,3),:))==1
      tri(i,:)=tri(i,[1 3 2]);
    end
  end
end

for i=j+1:length(xi)
  temp=con_hull(oD(oldhull,:),'i'); % form hull ignoring interior points
  hull=oldhull(temp); % return to original indices
  last=find(hull==(i-1)); % the sort makes each new pt on the hull
  % straigten out hull index, ccw starting at s_{i-1} and wrapping around
  hull=[hull([last:length(hull)]);hull([1:last])];
  % When we traverse the newest hull ccw from s_{i-1}=oD(last,:)=oD(hull(1),:)
  % and s_{i}=oD(i,:) is to the right then form a triangle
  for k=1:length(hull)-1
    if ccw(oD(hull(k),:),oD(hull(k+1),:),oD(i,:))== -1 % to the right
      tri=[tri; i hull(k) hull(k+1)]; % form cw traingle
    end
  end
  oldhull=hull;
  oldhull(length(oldhull))=i; % add current point for next check
end

% Dtri is tri in terms of original D indices
Dtri=reshape(xi(tri),size(tri,1),size(tri,2));

%Test to ensure we have all triangles where
n=size(oD,1);            % number of points (minus duplicates)
temp=con_hull(oD(oldhull,:),'i');
hull=xi(oldhull(temp)); % return to original indices
k=length(hull);          % # points on final hull
t=2*(n-1)-k;             % # of expected triangles
if t~=size(tri,1)
  % next should be error
  disp([' ']);disp('  Number of triangles is incorrect');
end

if pl==1  % To see whats going on
  plot(D(:,1),D(:,2),'.')
  hold on
  for N=1:size(D,1)
    text(D(N,1),D(N,2),int2str(N))
  end
  for i=1:size(tri,1)
    plot(oD(tri(i,[1:3,1]),1),oD(tri(i,[1:3,1]),2),'r')
  end
  hold off
end
```

C.4 Listing of MATLAB Function `incircle.m`

```
function inside=incircle(A,B,C,D)
% incircle - Used to determine if D is interior to the region of the plane
% that is bounded by the oriented circle ABC and lies to the left of it.
% In particular this implies that D is inside the circle ABC if the points
% A,B, and C define a counterclockwise oriented triangle and outside if
% they define a clockwise oriented one.
% All inputs must be distinct elements of R^2
% Interpetation of results
%      inside =  1,  D is inside the oriented circle ABC
%      inside =  0,  D is on the oriented circle ABC
%      inside = -1,  D is outside the oriented circle ABC
%
% see ref for developement
%  Guibas & Stolfi, ACM Tans. on Graphics 4(2):74-123, 1985 page 106
%
%     inside = incircle(A,B,C,D)
%
%  NAME:  incircle
%  LAST REVISION: 14 Sep 94  MatLab 4.2
%  Author: TOM KOBYLARZ  Air Force Institute of Tech WPAFB, OH
%          kobylar@afit.af.mil

if (nargin~=4),eval(['help incircle']), error('wrong number of input arguments.'); end

if size(A,2)~=2  % try to get elements in required format
  A=A';
end
if size(A,2)~=2  % check to see if ok now
  eval(['help incircle']);
  error('     A must be an element of R^2');
end
if size(B,2)~=2  % try to get elements in required format
  B=B';
end
if size(B,2)~=2  % check to see if ok now
  eval(['help incircle']);
  error('     B must be an element of R^2');
end
if size(C,2)~=2  % try to get elements in required format
  C=C';
end
if size(C,2)~=2  % check to see if ok now
  eval(['help incircle']);
  error('     C must be an element of R^2');
end
if size(D,2)~=2  % try to get elements in required format
  D=D';
end
if size(D,2)~=2  % check to see if ok now
  eval(['help incircle']);
  error('     D must be an element of R^2');
end

ordet=det([A sum(A.^2) 1;B sum(B.^2) 1;C sum(C.^2) 1;D sum(D.^2) 1]);
%if abs(ordet) <= 2*eps
if abs(ordet) <= 100*eps
  inside=0;  % try try minimize numerical error
else
  inside=sign(ordet);
end
```

C.5 Listing of MATLAB Function `ccw.m`

```matlab
function orientation=ccw(A,B,C)
% ccw - counterclockwise. Used to determine orientation of 3 points in R^2,
% positive when pt C lies to the left of the directed line segment AB
%
% Interpetation of results
%     orientation =  1,  Ordered points have a counterclockwise orientation
%     orientation =  0,  Ordered points are colinear
%     orientation = -1,  Ordered points have a clockwise orientation
%
% see refs for explaination and extensions to higher dimensions
% 1) Du, Ding-Zu and Hwanf, F., Computing in Euclidean Geometry,
%     World Scientific, 1992, QA447.C573 p. 210
% 2) Guibas & Stolfi, ACM Tans. on Graphics 4(2):74-123, 1985
%
%     orientation = ccw(A,B,C)
%
%  NAME:  ccw
%  LAST REVISION: 14 Sep 94  MatLab 4.2
%  Author: TOM KOBYLARZ  Air Force Institute of Tech WPAFB, OH
%          tkobylar@afit.af.mil

if (nargin~=3),eval(['help ccw']), error('wrong number of input arguments.'); end

if size(A,2)~=2  % try to get elements in required format
  A=A';
end
if size(A,2)~=2  % check to see if ok now
  eval(['help ccw']);
  error('     A must be an element of R^2');
end
if size(B,2)~=2  % try to get elements in required format
  B=B';
end
if size(B,2)~=2  % check to see if ok now
  eval(['help ccw']);
  error('     B must be an element of R^2');
end
if size(C,2)~=2  % try to get elements in required format
  C=C';
end
if size(C,2)~=2  % check to see if ok now
  eval(['help ccw']);
  error('     C must be an element of R^2');
end

ordet=det([A 1;B 1;C 1]);
%if abs(ordet) <= 2*eps
if abs(ordet) <= 100*eps
  orientation=0;  % try try minimize numerical error
else
  orientation=sign(ordet);
end
```

C.6 Listing of MATLAB Function con_hull.m

```
function edge=con_hull(z,s,p)
%con_hull  Find the Convex hull of z=(x,y), a set of at least 2 points in
%    R^2. Each row of z must be a 2-D vector.  s is a switch to determine
%    the ouput format, if s='i' indecies are returned instead of point values
%    If no output arguments are specified or p=1, a plot generated.
%    The returned vector edge, contains an ordered subset of z, by row
%    (index or value (default) depending on s), which generates a counter
%    clockwise oriented convex hull of z.  This must be noted when the results
%    are used for instance in a line integral.  An example use which generates
%    correct results for computing the area of the ccw hull of z when values
%    are returned is:  cover = area(edge(:,1),edge(:,2))
%
%      edge = con_hull(z,s,p)
%
%  NAME:  con_hull
%  LAST REVISION: 10 Feb 95  MatLab 4.2
%  Author: TOM KOBYLARZ  Air Force Institute of Tech WPAFB, OH
%            tkobylar@afit,af.mil


%  figure out input format and output requested by parameters provided
if nargout==0,pl=1;else,pl=0;end
if (nargin >3),eval(['help con_hull']), error('wrong number of input arguments.'); end
if nargin==0,eval(['help con_hull']),return; end
if nargin==1,s='p';end
if nargin==3,pl=p;end
if min(size(z))<2
  eval(['help con_hull']);
  error('       z must contain at least 2 sets of points (x,y)');
end
if size(z,2)~=2  % try to get z in required format
  z=z';
end
if size(z,2)~=2  % check to see if ok now
  eval(['help con_hull']);
  error('       z must contain 2 columns, each row a 2-D vector');
end

done=0;    % Hull is complete Flag


% Jump right to a point z(i) KNOWN to be on the boundary such as min(y) which
% assures 0 <= rays(:) <= pi. When ties in min y value occur we have a horizontal
% bottom edge. To properly terminate need to ID min(x) of tied y_mins

[z,yi]=sort_nd(z,[2,1],100);

dups=find(abs(diff(z(:,1)))<=100*eps & abs(diff(z(:,2)))<=100*eps);
if ~isempty(dups)
  disp(['       Duplicate Points Removed']);
  z(dups+1,:)=[];
  yi(dups+1)=[];
end
x=z(:,1);y=z(:,2);
i=1;                       % By means of the sorting

zt=[x-x(i) y-y(i)];         % translate to origin
rays=atan2(zt(:,2),zt(:,1));  % atan2 returns +/- pi
rays(i)=5;                  % avoid skewing min by translation
[minr,n1]=min(rays);
n1v=find(abs(rays-minr)<=1000*eps); % index of possible colinear segment
n1=n1v(1);
rays(i)=0;                  % restore ray
[maxr,n2]=max(rays);
```

```
% Once the edge is found should be able to stay on it
% By starting at min(y) ===>  0 <= rays(:) <= pi
% ===> z(n2,:) lies to the left of z(i,:) and z(n1,:) is to the right

edge=[i; n1];% safest to get only two points since this is always a corner
i=n1;  % set index to last point in edge
if pl==1,
 plot(z(:,1),z(:,2),'.')
 hold on
 plot(x(edge(1)),y(edge(1)),'g*') % The origin
 plot(x(i),y(i),'c+')                    % first ordered point index
 for N=1:size(z,1),text(z(N,1),z(N,2),int2str(yi(N))),end % number points
end

% Fix for when all points are colinear
if abs(maxr-minr) <= 1000*eps
 disp(' ');disp('    Points form a straight line')
 edge=1:length(yi)'; % works for vert & non-vert due to sort
 done=1;
end

%  Now that we're on the boundary use Jarvis' March (Gift Wrap Approach)

while done~=1 % find all points on the boundary
  zt=[x-x(i) y-y(i)];               % translate to origin
  rays=atan2(zt(:,2),zt(:,1));      % atan2 returns +/- pi
  rays(i)=5;                        % avoid skewing min by translation
  [minr,n1]=min(rays);
  rays(i)= -5;                      % avoid skewing max by translation
  [maxr,n2]=max(rays);
  rays(i)=0;                        % restore ray
  if maxr-minr > pi   % performing wrap correction to keep < 180 deg
    wrap=find(rays<0);
    rays(wrap)=rays(wrap) + 2*pi * ones(size(wrap),1);
    rays(i)=5;                      % avoid skewing min by translation
    [minr,n1]=min(rays);
    rays(i)=0;                      % restore ray
    [maxr,n2]=max(rays);
  end  % end of wrap correction

  n1v=find(abs(rays-minr)<=1000*eps); % index of possible colinear segment
  if length(n1v)>1
    n1v(find(n1v==i))=[];               % remove the current point if here
    side1=find(min(abs(n1v-i))==abs(n1v-i));
    if side1~=1,n1v=flipud(n1v);end % sort away from point
    n1=n1v(1);                        % adjacent point in this direction
  end
  n2v=find(abs(rays-maxr)<=1000*eps); % index of possible colinear segment
  if length(n2v)>1
    n2v(find(n2v==i))=[];               % remove the current point if here
    side2=find(min(abs(n2v-i))==abs(n2v-i));
    if side2~=1,n2v=flipud(n2v);end % sort away from point
    n2=n2v(1);                        % adjacent point in this direction
  end

  if edge(size(edge,1)-1) == n1     % maintain same direction
    i=n2;
    i=n2v(size(n2v,1));
    nv=n2v;
  else
    i=n1v(size(n1v,1));
    nv=n1v;
  end
  if pl==1,plot(x(nv),y(nv),'c+'),end
```

```
    edge=[edge; nv];                          % add pts to the edge

  if i==edge(1)  % We wrapped around and are done
    done=1;
    edge(size(edge,1))=[];  % remove origin
  end
  if length(edge) > length(x)+1
    done=1;
    disp('        Got an error here.  Too many vertices');
  end
end

if pl==1
  plot([x(edge);x(edge(1))],[y(edge);y(edge(1))],'r')
  hold off
end

% Now do the line Integral to find the enclosed area.
% Area returns positive for counter-clockwise orientation of edge
%cover=area(x(edge),y(edge))

if strcmp(s,'i')
  edge=yi(edge);  % convert back to original index
else
  edge=[x(edge),y(edge)];
end
```

C.7 Listing of MATLAB Function sort_nd.m

```
function [sorted,index]=sort_nd(z,order,tol)
% sort_nd      Sort the list of vectors z in ascending order where
%      the significance of elements is described in the vector order.
%      Each row of z is a vector and if order=[] the significance is
%      1:size(z,2).  NOTE: ELEMENTS WITHIN tol*EPS OF EACH OTHER ARE
%      CONSIDERED AND RETURNED AS IDENTICAL, the default is tol=100.
%      Example if elements in z are 2D z=[x,y;] then
%      sort_nd(z)=sort_nd(z,[1,2]) and the elements are sorted on x
%      with ties being sorted on y final ties are sorted by original
%      index in z. sort(z,[2,1]) sorts on y
%
%      [sorted,index]=sort_nd(z,order,tol)
%
%  NAME:   sort_nd
%  LAST REVISION: 14 Dec 94  MatLab 4.2
%  Author: TOM KOBYLARZ  Air Force Institute of Tech WPAFB, OH
%          tkobylar@afit,af.mil

if nargin==0,eval(['help sort_nd']),return; end
[m,n]=size(z);
if nargin==1,order=[1:n];end
if nargin<3, tol=100;end
if isempty(order),order=[1:n];end
if length(order)~=n | any(order>n),
  eval(['help sort_nd']), error('  Invalid significance order specified');
end

sorted=z;
index=[1:m]';

for i=n:-1:1
  [junk,in]=sort(sorted(:,order(i)));
  near=find(abs(diff(junk))<=tol*eps);        % catch numerics
  if ~isempty(near)
    for j=1:length(near),
      sorted(in(near(j)+1),order(i))=sorted(in(near(j)),order(i));
    end
    [junk,in]=sort(sorted(:,order(i)));       % sort on y again
  end
  sorted=sorted(in,:);
  index=index(in);
end
```

C.8 Listing of MATLAB Function fom_nl.m

```
function merit=fom_nl(t,output,cmd,pl)
% LAST REVISION: 3 Feb 95  improved cmd=0 usage
% Author: TOM KOBYLARZ  Air Force Institute of Tech WPAFB, OH
%
% fom_nl    Calculate the Classical Figures of Merit of an output
%           response (output) given the commanded input (cmd), both
%           of which must have the same time scale (t) unless cmd is
%           a scalar, in which case a step of value cmd with initial
%           condition output(1) is assumed. If cmd is omitted, zero
%           tracking error is assumed. Used to find a soft error norm
%           for evaluating nonlinear responses.
%
%     merit=fom_nl(t,output,cmd,pl)
%
%             If no left hand side argument is specified or pl=1,
%           the response is plotted indicating the figures of merit.
%             If a left hand argument is given, the result is a three
%           row matrix indicated below. The columns are zero padded
%           in case there are more extrema than FOMs (currently 7).
%
%                      | Mp    tp Te ts tr ref FV ... 0 ... |
%           merit =    | [overshoots]            ... 0 ... |
%                      | tstop [ratio]           ... 0 ... |
%
%     ref Commanded Step Value              FV  Final Value
%     tstop Duration of simulation          tp  Peak Time
%     ts  Actual 2% settling time to FV     tr  10% to 90% Rise Time
%     Mp  Peak Value at tp                  Te  Tracking error of FV
%     [overshoots] vector of percent overshoot at each local extrema prior to ts
%     [ratio] vector of overshoot ratios from one extrema to next


% figure out input format by number or parameters provided
if (nargin < 2) | (nargin > 4),
 eval('help fom_nl'),
 error('wrong number of input arguments.');
end
if nargin~=4,pl=0;end % default is no plot
% ensure we always start with row vectors
  if (min(size(t))>1), error('time argument must be a vector'); end
  if (size(t,2)>1), t=t'; end
  if (min(size(output))>1), error('response argument must be a vector'); end
  if (size(output,2)>1), output=output'; end
if (length(t)~=length(output)), error('time arguments (t,output) must match.'); end
npts=length(t);
if (nargin ==2),
    cmd=0*output + output(npts);    %assumes zero tracking error
    cmd(1)=output(1);               % assumes started from trim
end
if (length(cmd)==1),
    cmd=0*output + cmd;
    cmd(1)=output(1);               % assumes started from trim
end
if (length(t)~=length(cmd)),
  error('time arguments (t,cmd) must match or cmd must be a scalar.');
end
if nargout==0 | pl==1,DISP=1;else,DISP=0;end

IC=output(1);              % store initial condition
y=output-IC; clear output; % Translate output to zero
cmd=cmd-IC;                % Translate command to zero
if cmd(npts)<0            % Check for negative step and set flag
  negstep=1;
  cmd = -cmd;            % make positive to obtain FOMs
```

```
  y = -y;
end
tstop=t(npts);
FV=y(npts);                     % Final Value - should be >0 except very small commands
ref=cmd(npts);                  % Commanded step strength

%  Check for dead time on first i points when y(i)=y(1)
dti=2;
while y(1) == y(dti),
  if dti==npts,
    if DISP==1
      disp('       !!!!!!!!!!!!!!    No deviation from trim value    !!!!!!!!!!')
    end
    merit=zeros(3,7);
    return
  end
  dti=dti+1;
end
% check initial deviation from trim for non minimum phase responses for STEP INPUT
if (y(dti)-y(dti-1))*(cmd(2)-cmd(1)) <0  % assumes monotonic cmd
  if DISP==1
    disp('            Non-Minimum Phase Response')
  end
  nonmin=1; % flag for non-minimum phase response
  % !!!!  needs to be expanded when NMP response has cost
else
  nonmin=0;
end

[Mp,tpi]=max(y);                % gives index of 1st occurrence of max
tp=t(tpi);                      % assumes max value is unique may cause problems
Te=100*abs(FV-ref)/ref;         % Steady state tracking error percent
Os=100*(Mp-FV)/FV;              % Percent overshoot of FV at Mp

firstdif=diff(y([dti-1:npts]));     % look for extrema or inflextion after initial dead time
                                    % when curve is FLAT between 2 points 1st diff=0
secdif=diff(sign(firstdif));        % since sign=(-1|0|1) then secdif=(-2|-1|0|1|2)
                                    % look for mins(secdef=2) and maxs(secdef=-2)
                                    % but secdif=(-1|0|1) may be min/max/inflex
local=find(abs(secdif)==2)+dti-1;   % since sign=0 when firstdif=0
flat=find(abs(secdif)==1)+dti-1;    % adjust indices to y

if length(flat)==1              % fix for step looking things
  local=[local;flat];
  flat=[];
end

if (~isempty(flat))             % Flat spots need extra work
  if rem(length(flat),2)==1     % since true flats come in pairs
    flat(length(flat),:)=[];    % remove last row if odd
  end
  squeeze=[flat,secdif(flat-dti+1)]
  for i=1:2:length(flat)-1                        % average the index to find middle
    squeeze(i,1)=fix((squeeze(i,1)+squeeze(i+1,1))/2);
    squeeze(i,2)=squeeze(i,2)+squeeze(i+1,2);     % add the 2 about the flat spot
    squeeze(i+1,:)=squeeze(i,:);
  end
  i=[1:2:length(flat)-1];
  squeeze=squeeze(i,:);                           %remove redundant rows
  squeeze=squeeze(find(abs(squeeze(:,2))~=0),:); %remove inflextion points
  flat=squeeze(:,1);
end

extremi=sort([local;flat]);     % order local extremum indecies
extrem=[t(extremi),y(extremi)]; % extract local extremum & time
```

C-17

```
above=min(find(abs(y - FV) < abs(.02*FV)));   % first time enters 2% band
below=max(find(abs(y - FV) > abs(.02*FV)));   % last time outside 2% band
tsi=max(above,below);                          % for monotonic ?
tsettle=t(tsi);
if isempty(tsettle)  % catch, probably unstable response
  tsettle=inf;
end

if size(extrem,1) > 1
  extrem=extrem(find(extrem(:,1)<tsettle),:); % ignore extremum in tail
end

if isempty(extrem)    % to cover monotonic responses
  extrem = [tp Mp];
end

if nonmin==1 % assumes one non-min phase excursion below zero extrem(1,:)
  if extrem(1,2)>0
    disp(' ');disp('     !! error in tracking non-min phase extrema');disp(' ')
  else
    if size(extrem,1)>1
      extrem(1,:)=[];    % currently no fom specified, remove from list
    end
  end
end

if extrem(1,:) ~= [tp Mp]
  if DISP==1
    disp(['          Peak overshoot is not first Maximum'])
  end
  badpeak=1;    % flag for maybe bad response, compare with nonmim
end

overshoots=100*(extrem(:,2)-FV)/FV; % percent error at each local extremum
overshoots=overshoots';


for i=1:length(overshoots)-1     % find ratios of consecutive extrema
  ratio(i)=overshoots(i+1)/overshoots(i);
end

tr1i=1;tr2i=tpi;   % initialize just incase problems with rise time
for i = 1:tpi                    % find rise time
  if y(i) < (.1 * FV), tr1i = i; end
  if y(i) < (.9 * FV), tr2i = i; end
end
tr1=t(tr1i);ytr1=y(tr1i);
tr2=t(tr2i);ytr2=y(tr2i);
tr=tr2-tr1;

merit=zeros(3,max(7,length(overshoots)));
merit(1,[1:7])=[Mp tp Te tsettle tr ref FV];
merit(2,[1:length(overshoots)])=overshoots;
merit(3,[1:length(overshoots)])=[t(npts) ratio];

%    Pretty output if merit not specified

if DISP==1,

  Mps=num2str(Mp);
  Oss=[num2str(Os),' %'];
  if ref==0
    Tes='Undefined, merit returns Inf';
  else
```

```
   Tes=[num2str(Te),' %'];
  end
  FVs=num2str(FV);
  refs=num2str(ref);
  if isempty(ratio)
  Maxratios='undefined';
  else
  Maxratios=num2str(ratio(find(abs(ratio)==max(abs(ratio)))));
  end
  if tr < tp & tr~=0
    trs=num2str(tr);
  elseif (tr==tp | tr==0)
    trs=['poorly defined'];
  else
    trs=['very large'];
  end
  if tp < tstop
    tps=num2str(tp);
  else
    tps=['very large'];
  end
  if tsettle < tstop
    tsettles=num2str(tsettle);
  else
    tsettles=['very large'];
  end

  if nargout==0
    disp(['Figures of merit'])
    disp([' '])
    disp(['rise time              =   ',trs])
    disp(['peak time              =   ',tps])
    disp(['settling time          =   ',tsettles])
    disp(['peak value             =   ',Mps])
    disp(['final value            =   ',FVs])
    disp(['commanded value        =   ',refs])
    disp(['initial condition      =   ',num2str(IC)])
    disp(['Percent Overshoot      =   ',Oss])
    disp(['Max ratio overshoot    =   ',Maxratios])
    disp(['Percent Track error    =   ',Tes])
  end

  plot(t,[y cmd])            % System Response and command
  hold on
  plot([tr1,tr1],[0,ytr1],'c:',[tr2,tr2],[0,ytr2],'c:',[tr1,tr2],[ytr1,ytr1],'c:')  % tr Lines
  plot([tp,tp],[0,Mp],'m:',[0,tp],[Mp,Mp],'m:')   % Peak Time Lines
  plot([tsettle,tsettle],[0,FV],'w:',[0,t(npts)],[FV,FV],'r:')% FV and ts Lines
  for i=1:size(extrem,1)        % Indicate local extremum
    plot([extrem(i,1),extrem(i,1)],[FV,extrem(i,2)],'g')
  end
  title(['Figures of Merit: Response Translated to Trim Condition of ',num2str(IC)])
  ylabel('Response'), xlabel('time')
  text(.03*t(length(t)),0.95*Mp,['Mp = ',Mps])
  text(1/4*(tr2+3*tr1),1.3*(0.1*FV),['tr = ',trs])
  text(0.9*tp,0.5*Mp,['tp = ',tps])
  text(0.9*tsettle,0.25*FV,['ts = ',tsettles])
  text(0.9*tsettle,0.8*FV,['local extrema'])
  text(0.9*tsettle,0.7*FV,[sprintf(' %5.1f ',overshoots)])
  if ~isempty(ratio),text(0.9*tsettle,0.5*FV,['ratio of extrema']),end
  if ~isempty(ratio),text(0.9*tsettle,0.4*FV,[sprintf(' %5.2f ',ratio)]),end
  hold off
end

return
```

C.9   Listing of MATLAB Function fuz_cost.m

```
function cost=fuz_cost(merit,display,weights,specifications)
%fuz_cost    Calculate the cost of the response with figures of merit
%            contained in the 3 row (zero padded) matrix merit
%            obtained from the m file  'fom_nl.m'  with form
%
%                       | Mp tp Te ts tr ref FV ... 0 ... |
%            merit =  | [overshoots]          ... 0 ... |
%                       | tstop [ratio]         ... 0 ... |
%
%    ref: Commanded Step Value               FV:  Final Value
%    tstop: Duration of simulation           tp:  Peak Time
%    ts:  Actual 2% settling time to FV      tr:  10% to 90% Rise Time
%    Mp:  Peak Value at tp                    Te:  Tracking error of FV
%    [overshoots]: vector of percent overshoot at each local extrema prior to ts
%    [ratio]: vector of overshoot ratios from one extrema to next
%
%        Call after using merit=fom_nl(t,y,cmd) as
%
%     cost=fuz_cost(merit,display,weights,specifications)
%
%        display: A switch to display violations when display=1 (default is 0)
%        weights: A vector of weights to place relative importance upon
%                 certain elements of the specification vector, defaults to
%                 weights=[1 1 5 1 1 1 1 1];
%specifications:  The vector containing max allowable specs for each figure of
%                 merit. If specs is not defined the default is used.  If only
%                 certain elements of specs are desired to be changed this is done
%                 by using a string to reset the elements separated by semicolons.
%                 i.e. use 'specs(2)=1;specs(5)=3;' to change the first and fifth
%                 elements to 1 and 3 respectively.
%                 See .m file code for use of specs in the cost function.
%
%                       1     2    ·  3            4      5      6    7    8
%           specs = [minMp maxMp max#overshoots maxratio maxerror maxtp maxts maxtr]
%                   where the elements conform to:
%       1,2:  fraction of normalized step strength ie .98 and 1.3 for -2%/+30% overshoot
%         3:  integer                                4:  positive real decimal
%         5:  percent error ie 2 for 2% default,  6-8:  seconds
%
%  LAST REVISION: 26 Jan 95 Handle trim conditions v 4.2c and slope for unstable
%                  9 Nov 94 define interactive mode to echo violations
%                  1 Aug 94  MatLab 4.0
%  Author: TOM KOBYLARZ  Air Force Institute of Tech WPAFB, OH

% figure out input format by number or parameters provided
if (nargin==0 | nargin >4),
 eval('help fuz_cost'),
 error('wrong number of input arguments.');
end
if (size(merit,1)~=3), error('merit must have form returned from fom_nl.m'); end
tstop=merit(3,1);
if nargin == 1, display=0; end
if nargin < 3,
  weights=[];
end
specs=[.98  1.25  4  0.4  1.0  tstop  5.0  tstop];
if nargin == 4,
  eval(specifications);
end

merit(3,1)=0;   % to simplfy code below

if isempty(weights),
```

```
  weights=[1 1 5 1 1 1 1 1];
end
penalty=0*weights;
% ensure we always start with correct vectors
  if (min(size(specs))>1), error('specifications must be a vector'); end
  if (min(size(weights))>1), error('weights must be a vector'); end
  if (length(weights)~=8), error('weights must be an 8 element vector'); end

%  add a large cost and return if peak value is Inf or NaN
% slope is provided by 10000*(1-tp/tstop) ie time to instabilty
if ( isnan(merit(1,1)) | isinf(merit(1,1)) )
  cost=10000 + (1-(merit(1,2)/tstop))*10000;
  return
end

%  Return zero cost and return if merit returned a valid trim case
if any(any(merit))==0
  cost=0;return
end

% Calculate penalties for deviating from spec
violations=penalty; % initialize flags to display violations

  if merit(1,1) < specs(1)*merit(1,6)
    penalty(1)= (specs(1)*merit(1,6)-merit(1,1));
    violations(1)=1;
  end

  if merit(1,1) > specs(2)*merit(1,6)
    penalty(2)= (merit(1,1)-specs(2)*merit(1,6));
    violations(2)=1;
  end

  if length(find(merit(2,:))) > specs(3)
    penalty(3)= (length(find(merit(2,:))) - specs(3));
    violations(3)=1;
  end

  if max(abs(merit(3,:))) > specs(4)
    penalty(4)= (max(abs(merit(3,:))) - specs(4));
    violations(4)=1;
  end

  if abs(merit(1,3)) > abs(specs(5))
    penalty(5)= (abs(merit(1,3)) - abs(specs(5)));
    violations(5)=1;
  end

  if merit(1,2) > specs(6)
    penalty(6)= (merit(1,2) - specs(6));
    violations(6)=1;
  end

  if merit(1,4) > specs(7)
    penalty(7)= (merit(1,4) - specs(7));
    violations(7)=1;
  end

  if merit(1,5) > specs(8)
    penalty(8)= (merit(1,5) - specs(8));
    violations(8)=1;
  end

  if min(penalty) < 0
    error(['negative cost element, there is a problem in the code'])
```

C-21

```
    end

penalty;
cost=sum(weights.*penalty);

% Build up display of violations if requested
if display==1,
  if max(violations)==1;
    dispMAT=str2mat([' '],['      !!!  VIOLATED the following specifications  !!!']);
  else
    dispMAT=str2mat([' '],['      All specifications satisfied']);
  end
  if violations(1)==1;
    dispMAT=str2mat(dispMAT,['                min Mp spec: ',num2str(specs(1)),' of final value with ',num2str(merit(1,1)/meri
  end;
  if violations(2)==1;
    dispMAT=str2mat(dispMAT,['                max Mp spec: ',num2str(specs(2)),' of final value with ',num2str(merit(1,1)/meri
  end;
  if violations(3)==1;
    dispMAT=str2mat(dispMAT,['                max Number of oscillations: ',num2str(specs(3)),' with ',num2str(length(find(mer
  end;
  if violations(4)==1;
    dispMAT=str2mat(dispMAT,['                max Ratio of oscillations: ',num2str(specs(4)),' with ',num2str(max(abs(merit(3,
  end;
  if violations(5)==1;
    dispMAT=str2mat(dispMAT,['                max Percent Tracking Error spec of ',num2str(specs(5)),'% with ',num2str(merit(1
  end;
  if violations(6)==1;
    dispMAT=str2mat(dispMAT,['                max Peak Time spec of ',num2str(specs(6)),' secs with ',num2str(merit(1,2))]);
  end;
  if violations(7)==1;
    dispMAT=str2mat(dispMAT,['                max Settling Time spec of ',num2str(specs(7)),' secs with ',num2str(merit(1,4)]
  end;
  if violations(8)==1;
    dispMAT=str2mat(dispMAT,['                max Rise Time spec of ',num2str(specs(8)),' secs with ',num2str(merit(1,5))]);
  end
  disp(dispMAT)
end

return
```

C.10 Listing of MATLAB Function find_2dv.m

```matlab
function var=find_2dv(D,A,cm)
%find_2dv  Find the variance parameters var for the membership functions
%    centered at D=[x;y] to provide cover of the 2D Universe of Discourse
%    such that no point has a cross membership greater than cm (0< cm <1).
%    The default cross membership is cm=.001 if not specified. D is a
%    vector of 2 rows and at least 3 columns (points). Each column of D
%    must be a unique 2-D vector. If D is colinear artifical bounds are
%    used to obtain finite solutions. A, the adjacency matrix as returned
%    from A=delaunay(D). If A is not specified or empty it is calculated.
%    If no output arguments are specified a plot is made.
%
%    var=find_2dv(D,A,cm)
%
% NAME:  find_2dv
% LAST REVISION: 15 Feb 95  Ignore Inf results as long as one valid max found
%                 8 Feb 95  Allowed for colinear and modified/invalid A
%                 1 Nov 94  MatLab 4.2
% Author: TOM KOBYLARZ  Air Force Institute of Tech WPAFB, OH
%         tkobylar@afit.af.mil


% figure out input format and output requested by parameters provided
if nargout==0,pl=1;else,pl=0;end
if (nargin >3),eval(['help find_2dv']), error('wrong number of input arguments.'); end
if nargin==0,eval(['help find_2dv']),return; end
npts=size(D,2);
if size(D,1)~=2              % check to see if D in required format
  eval(['help find_2dv']);
  error('       D must contain 2 rows, each column a 2-D vector');
end
if npts < 3
  eval(['help find_2dv']);
  error('       D must contain at least 3 sets of points [x;y]');
end
if nargin==1,
  A=delaunay(D);            % Find Adjacency matrix for Delaunay Diagram
end
if isempty(A),
  A=delaunay(D);            % Find Adjacency matrix for Delaunay Diagram
end
if size(A,2)~=npts | size(A,1)~=npts
  eval(['help find_2dv']);
  error('       A must be the adjacency matrix as returned from A=delaunay(D)');
end
if nargin<3,cm=.001;end
if cm<=0 | cm>=1            % check for valid cross membership
  eval(['help find_2dv']);
  error('       Cross Membership must be in the range 0 < cm < 1');
end


% SOLVE THE CONSTANT MEMBERSHIP ELLIPSE, ie maximize the area of a concentric
% ellispe. centered at the point, closest other point on the ellipse and all
% other points lie on or outside the ellipe.  after the semi-major/minor axis
% are determined, then solve for the required variance paramters such that
% this ellipse reprsents a contour of the desired cross memebership.
% use x^2*xstar+y^2*ystar=1 for ellipse ==> xstar=1/a^2 ystar=1/b^2
% The following assumes D is 2 by n

var=0*D;

for N=1:npts
  cpt=D(:,N);
  neigh=find(A(N,:));
  opts=D(:,neigh);
```

```
%  include all points in constraints but use max a&b
%  found from neighbors when they yield a finite solution
normpts=D-cpt(:,ones(1,npts));           % center points at cpt
normpts(:,N)=[];                         % remove cpt from consideration
[junk,i]=min(sqrt(sum(normpts.^2)));     % find closest point
closept=normpts(:,i);                    % closest pt is on ellipse
slope=closept(2)/closept(1);             % for use in unconstrained pts
normpts(:,i)=[];                         % remove closest from consideration

% form the constraints on a & b for finite solutions
amin=min(abs(cpt(1)-opts(1,:)));  % find min/max projections to neighbors
amax=max(abs(cpt(1)-opts(1,:)));
bmin=min(abs(cpt(2)-opts(2,:)));
bmax=max(abs(cpt(2)-opts(2,:)));

% when induced constriants are from closest point we may still be unconstrained
% catch for perp to axis and points with a single neighbor
if amax < eps                    % all neighboors aligned in y.
  if max(abs(cpt(1)-D(1,:))) > eps
    amax=max(abs(cpt(1)-D(1,:)));
    disp(['  !! Suspect Adjacency Matrix, check neighboors of point ',int2str(N)])
  else                           % all points colinear in y.
    amax=abs(max(D(2,:))-min(D(2,:)));% artifical bound of span of y
  end
end
if bmax < eps                    % same as for amax but aligned in x
  if max(abs(cpt(2)-D(2,:))) > eps
    bmax=max(abs(cpt(2)-D(2,:)));
    disp(['  !! Suspect Adjacency Matrix, check neighboors of point ',int2str(N)])
  else                           % all points colinear in x.
    bmax=abs(max(D(1,:))-min(D(1,:)));
  end
end
% catch for colinear pts not perp to axis
if amin==amax & amin==abs(closept(1,1))  % need to relax x constriant
  np=sort(abs(normpts(1,:)));            % sorted queque of x projections
  i=1;flag=0;
  while flag==0
    if np(i) > amax
      amax=np(i);
      flag=1;
    else
      i=i+1;
      if i > length(np)
        flag=1;
        amax=abs(max(D(1,:))-min(D(1,:)));
      end
    end
  end
end
if bmin==bmax & bmin==abs(closept(2,1))  % need to relax y constriant
  np=sort(abs(normpts(2,:)));            % sorted queque of y projections
  i=1;flag=0;
  while flag==0
    if np(i) > bmax
      bmax=np(i);
      flag=1;
    else
      i=i+1;
      if i > length(np)
        flag=1;
        bmax=abs(max(D(2,:))-min(D(2,:)));
      end
    end
```

```
       end
     end

     ab_bds=[amin amax;bmin bmax]; % artifical bds must not violate inequalites found later
     xy_bds=1./fliplr(ab_bds.^2); % = [xmin xmax;ymin ymax]
     xy_const=[0 inf;0 inf];
     xy_lim=xy_const;

     % Find constraints such that no other point interrior to ellipses
     if abs(closept(2))>eps
       alphax=[normpts(1,:).^2 - (closept(1)/closept(2))^2*normpts(2,:).^2]';
       betax=[1 - (normpts(2,:)/closept(2)).^2]';
       neg=find(alphax<-eps);pos=find(alphax>eps);
       if ~isempty(pos),xy_const(1,1)=max(betax(pos)./alphax(pos));end
       if ~isempty(neg),xy_const(1,2)=min(betax(neg)./alphax(neg));end
     else
       xy_const(1,:)=abs([closept(1) closept(1)]);
     end
     if abs(closept(1))>eps
       alphay=[normpts(2,:).^2 - (closept(2)/closept(1))^2*normpts(1,:).^2]';
       betay=[1 - (normpts(1,:)/closept(1)).^2]';
       neg=find(alphay<-eps);pos=find(alphay>eps);
       if ~isempty(pos),xy_const(2,1)=max(betay(pos)./alphay(pos));end
       if ~isempty(neg),xy_const(2,2)=min(betay(neg)./alphay(neg));end
     else
       xy_const(2,:)=abs([closept(2) closept(2)]);
     end

     % combine these constraints to generate resonable ellipses based upon
     % xy_bds. NOTE: must ensure the analytical constraints of xy_const
     % are not violated in the process.
     xy_lim(:,1)=[min(max(xy_const(:,1),xy_bds(:,1)),xy_const(:,2))];
     xy_lim(:,2)=[max(min(xy_const(:,2),xy_bds(:,2)),xy_const(:,1))];

     % Solve for the ellipse based on amax
     xstar=xy_lim(1,1);
     astar=1/sqrt(xstar);
     ystar=(1-xstar*closept(1)^2)/closept(2)^2;
     bstar=1/sqrt(ystar);
     circum_area=[astar bstar astar*bstar];

if pl==1
% generate points on amax ellipse to plot
  x= -astar+eps:astar/50:0;
  ye=sqrt((1-x.^2*xstar)/ystar);
  x=[x fliplr(-x) -x fliplr(x)];
  ye=[ye fliplr(ye) -ye fliplr(-ye)];
end

     % Solve for the ellipse based on bmax
     ystar=xy_lim(2,1);
     bstar=1/sqrt(ystar);
     xstar=(1-ystar*closept(2)^2)/closept(1)^2;
     astar=1/sqrt(xstar);
     circum_area=[circum_area; astar bstar astar*bstar];

if pl==1
% generate points on bmax ellipse to plot
  y= -bstar+eps:bstar/50:0;
  xe=sqrt((1-y.^2*ystar)/xstar);
  y=[y fliplr(-y) -y fliplr(y)];
  xe=[xe fliplr(xe) -xe fliplr(-xe)];

  plot(normpts(1,:),normpts(2,:),'o')
  hold on
```

```
    plot(opts(1,:)-cpt(1),opts(2,:)-cpt(2),'x')
    plot(x,ye,'r',xe,y,'c')
    plot(0,0,'g+',closept(1),closept(2),'wx')
    title(['Point ',int2str(N)])
    hold off
%   disp('    Hit any key to continue');  pause
    axis('equal')
    pause(.1)
  end

    % choose the largest and find corresponding var parameters
    % just incase still poorly constrained zero that result
    still_bad=find(isinf(circum_area(:,3))|isnan(circum_area(:,3)));
    if ~isempty(still_bad)
      circum_area(still_bad,3)=0; % as longs as one is finite we're ok
    end
    best=find(max(circum_area(:,3))-circum_area(:,3)<100*eps);
    if length(best)==2  % select better orientation in a tie
      ratio=circum_area(:,1)./circum_area(:,2);
      if abs(slope)>=1
        best=find(max(ratio)==ratio);
      else
        best=find(min(ratio)==ratio);
      end
    end
    if length(best)==2,best=1;end  % incase their identical

    var(1,N)=cros_mem(0,circum_area(best,1),cm);
    var(2,N)=cros_mem(0,circum_area(best,2),cm);

end    % next N

if min(min(var))<=0,
  disp(['    var has an element <=0 something is wrong']),break
end

if pl==1
  MF=[D',var'];
  gauss_2d('plot_it',MF,'s',[1 0 0]);
%  gauss_2d('plot_it',MF,'s',[1 0 0 [100*cm 10*cm cm]]);
%  u=[min(D(1,:)) max(D(1,:)) min(D(2,:)) max(D(2,:))];
%  w=[u(2)-u(1) u(4)-u(3)];
%  axis(u + .2*[-w(1) w(1) -w(2) w(2)]);
end
```

C.11 Listing of MATLAB Function `cros_mem.m`

```
function var=cros_mem(center,other_pts,weight)
%       var = cros_mem(center,other_pts,weight)
%
% Used to find the variance paramters for Gaussian membership functions
% centered at other_pts such that their membership value is equal to
% weight when evaluated at the point center.  That is solve
%       weight=exp(-1/2*(center-other_pts)^2/var)
% for var.  Requires that :  0 < weight < 1

if (nargin~=3),eval(['help cros_mem']), error('wrong number of input arguments.'); end
if (weight<=0 | weight>=1),
  eval(['help cros_mem']), error('Invalid weight specified');
end

var = -.5 * (center-other_pts).^2/log(weight);
```

Appendix D. Support Data for Chapter IV

D.1 Increase in Cover by Scheduler: Nonlinear System
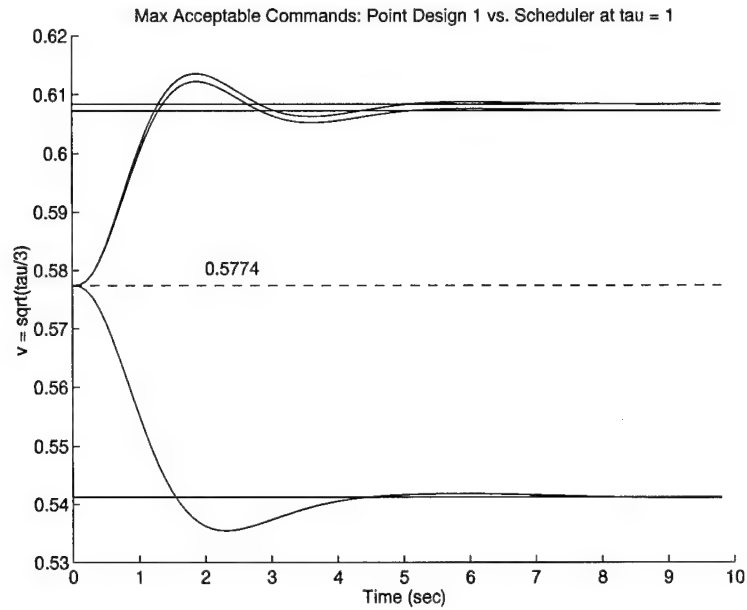
Figure D.1   Increase in Slewing Capability from Point Design 1
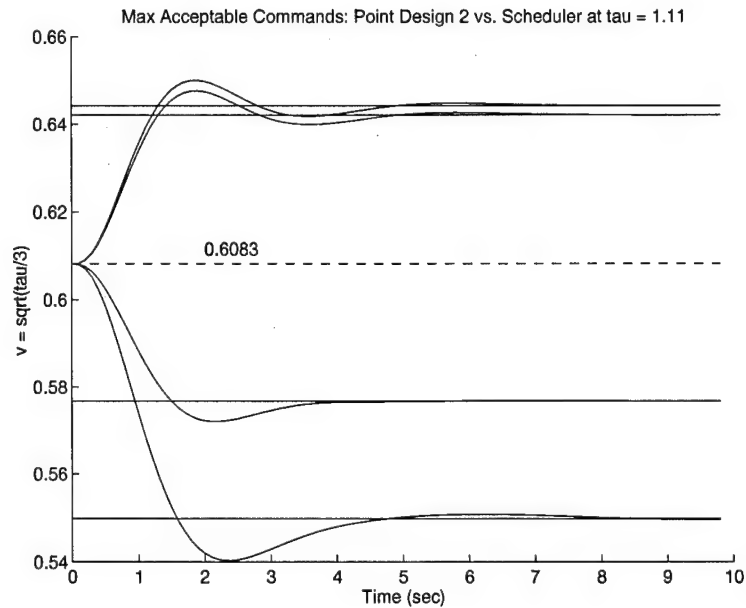
Figure D.2   Increase in Slewing Capability from Point Design 2
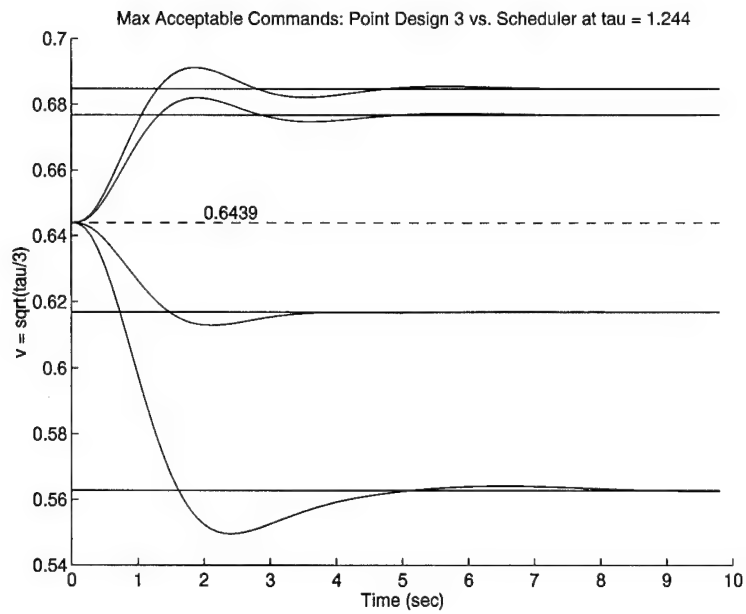
Figure D.3   Increase in Slewing Capability from Point Design 3
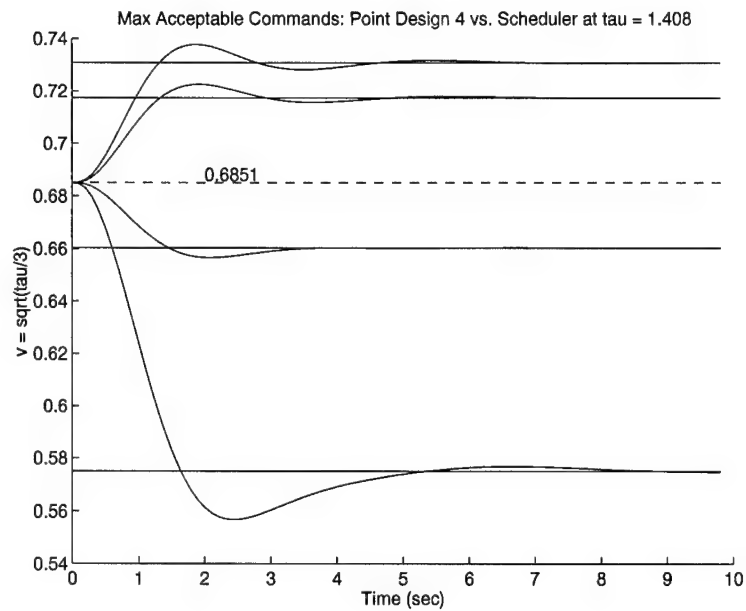
Figure D.4   Increase in Slewing Capability from Point Design 4
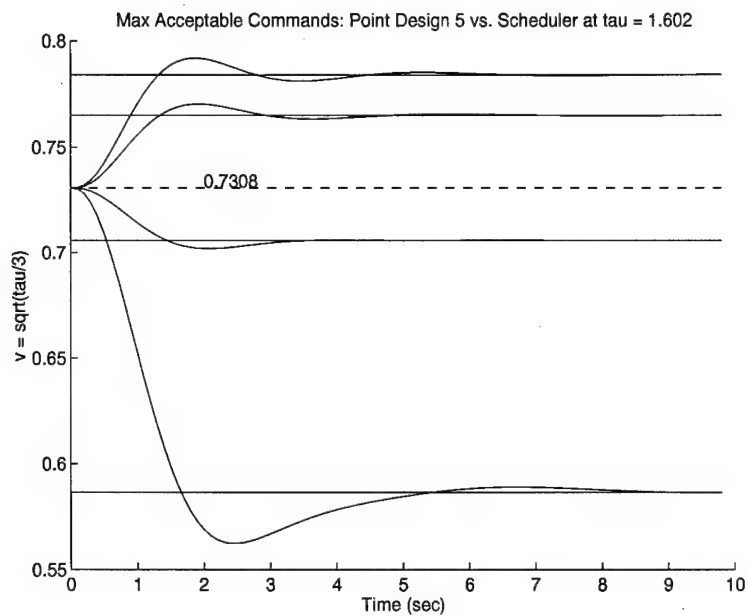
Figure D.5   Increase in Slewing Capability from Point Design 5
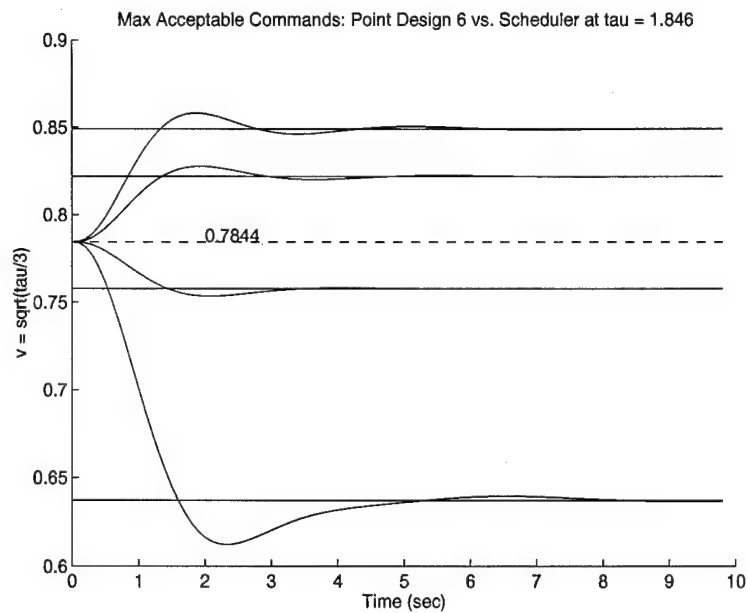
Figure D.6   Increase in Slewing Capability from Point Design 6
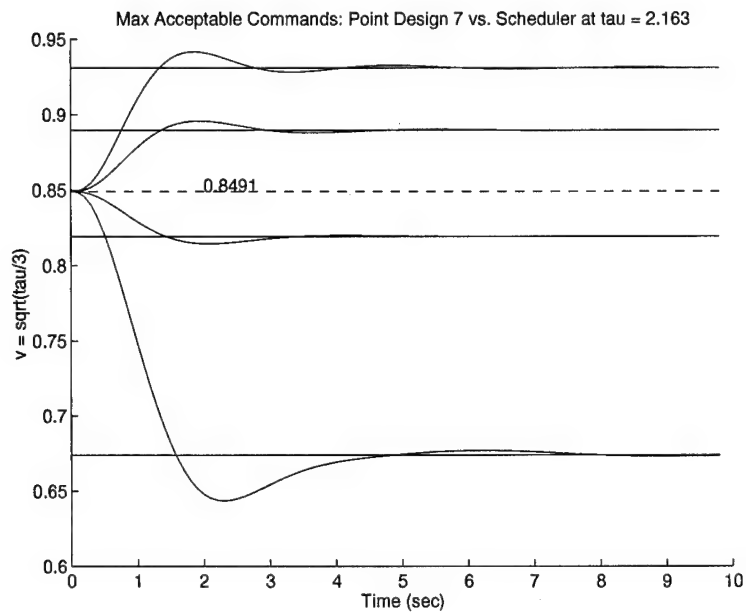
Figure D.7   Increase in Slewing Capability from Point Design 7
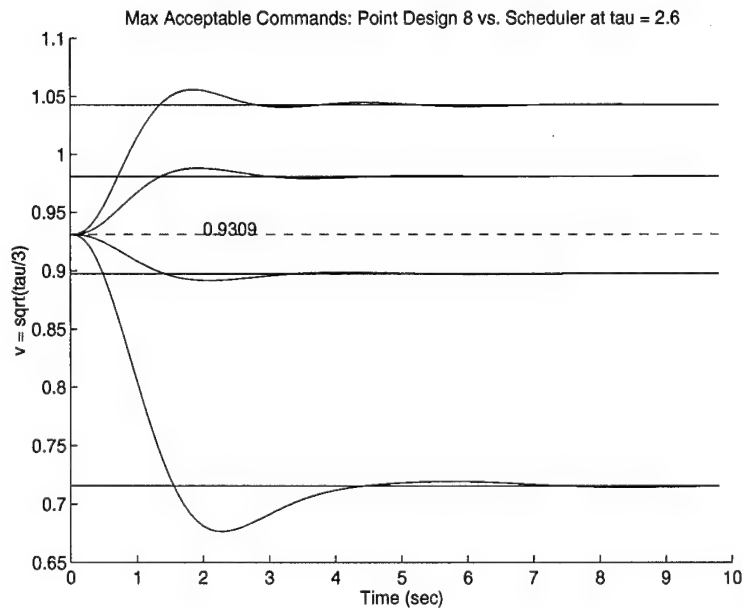
Figure D.8   Increase in Slewing Capability from Point Design 8

D-4
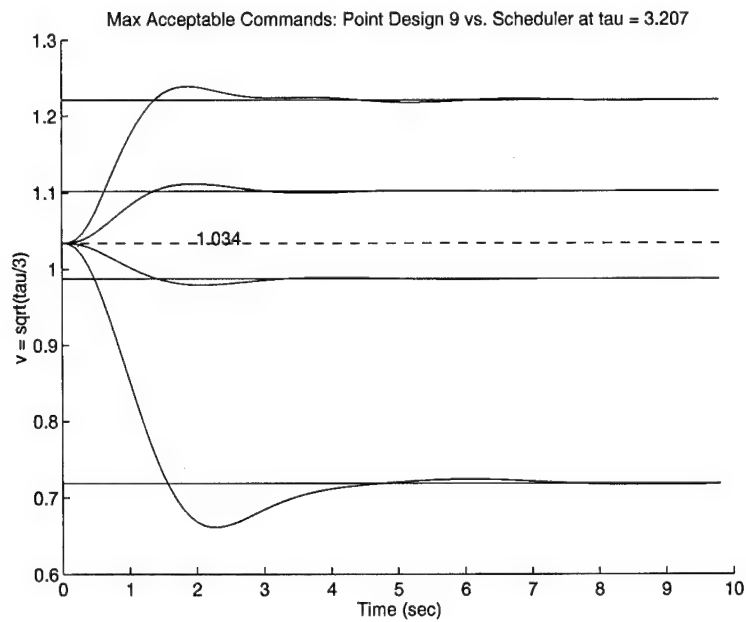
Figure D.9   Increase in Slewing Capability from Point Design 9
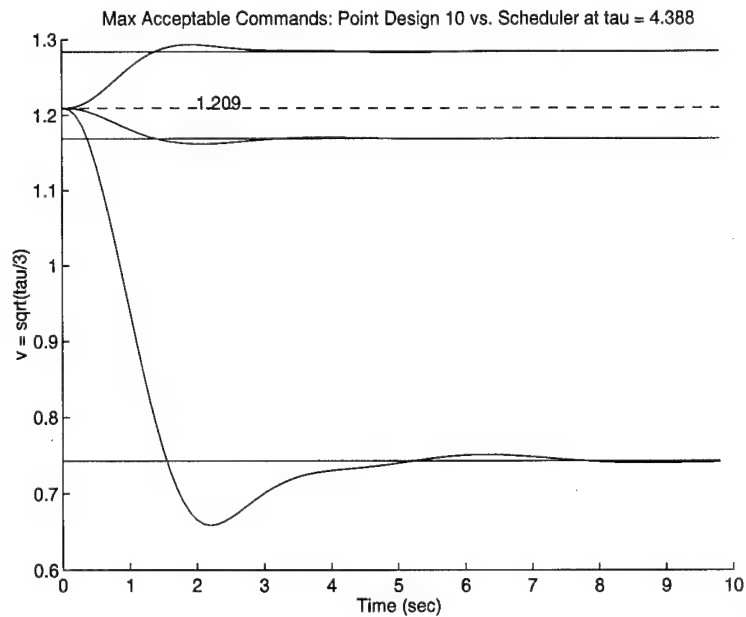
Figure D.10   Increase in Slewing Capability from Point Design 10

D-5

D.2 Constraint Surface Plots of Point Controllers: Nonlinear System

Constraint Surface for Point Design 1 at tau = 1 or v = 0.5774
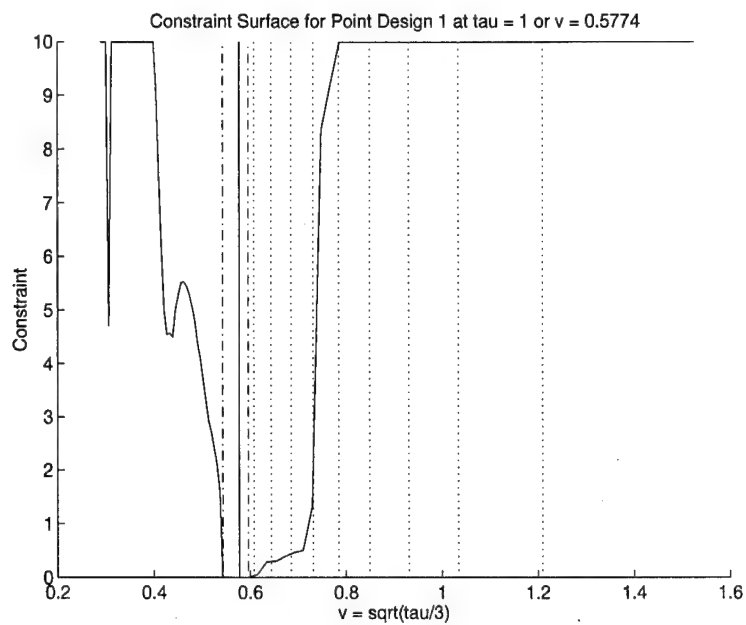
Figure D.11   Constraint Surface for Point Controller 1

Constraint Surface for Point Design 2 at tau = 1.11 or v = 0.6083
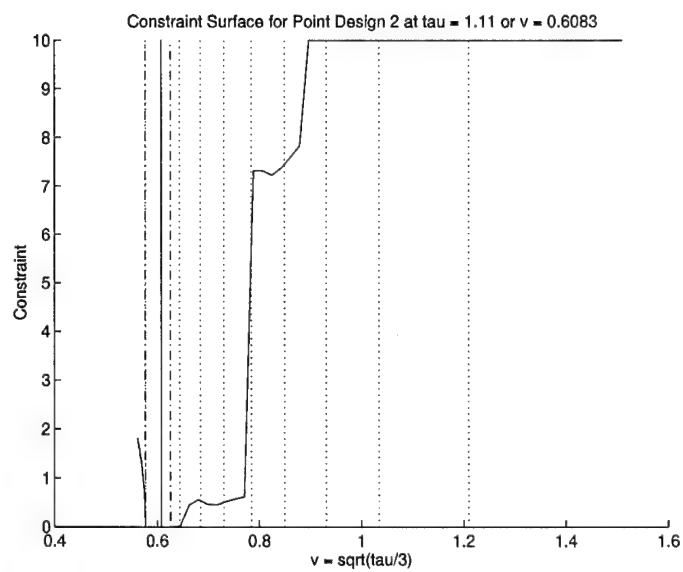
Figure D.12   Constraint Surface for Point Controller 2
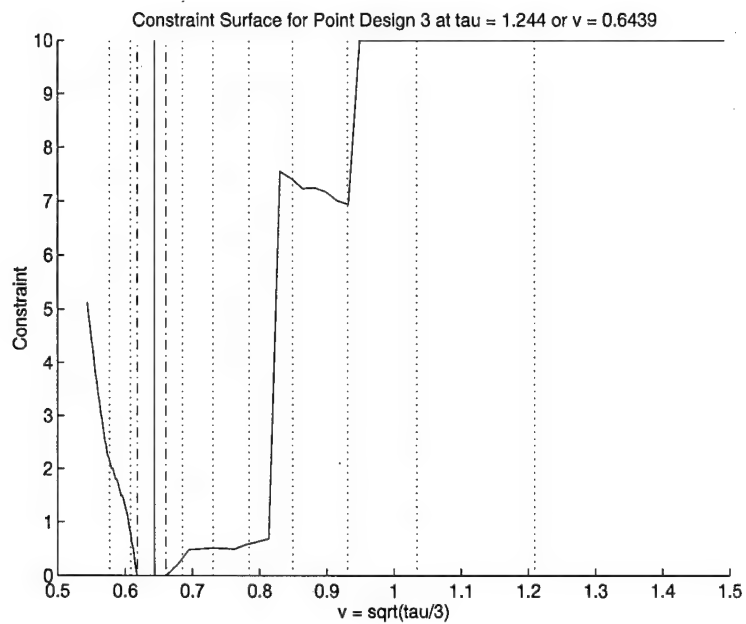
Figure D.13   Constraint Surface for Point Controller 3
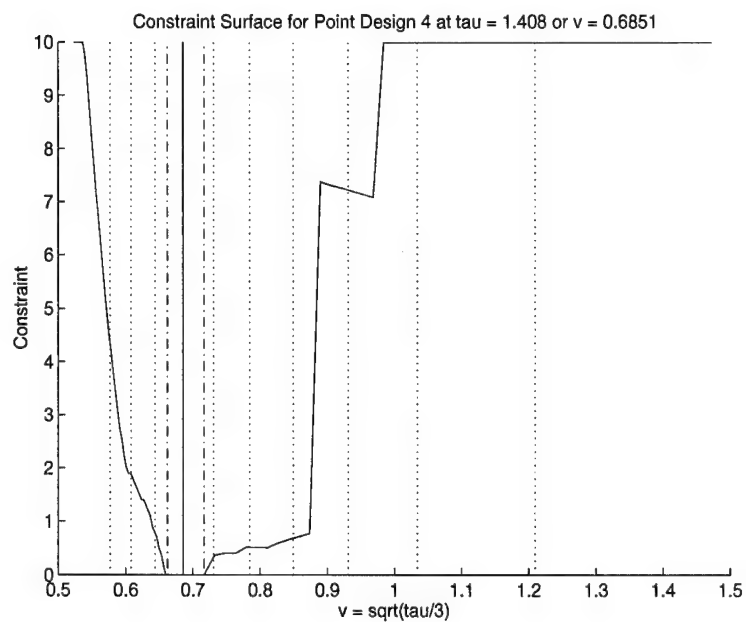
Figure D.14   Constraint Surface for Point Controller 4
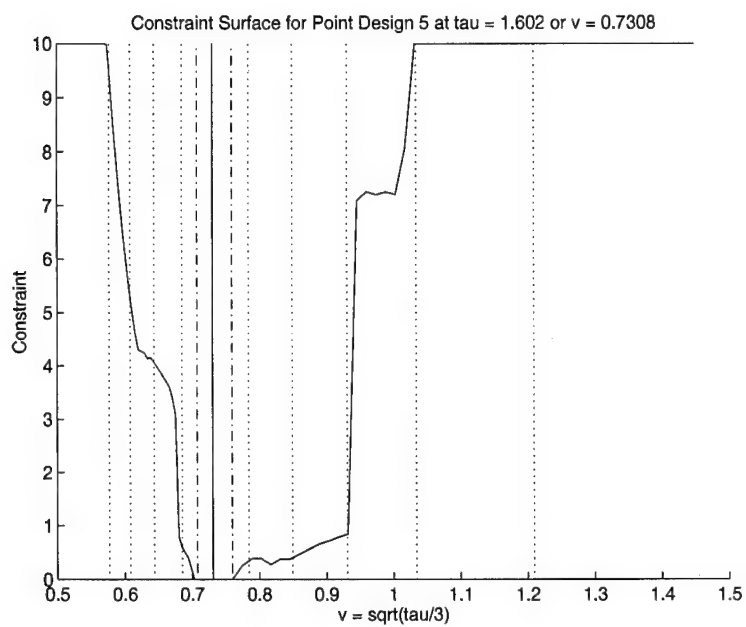
Figure D.15   Constraint Surface for Point Controller 5
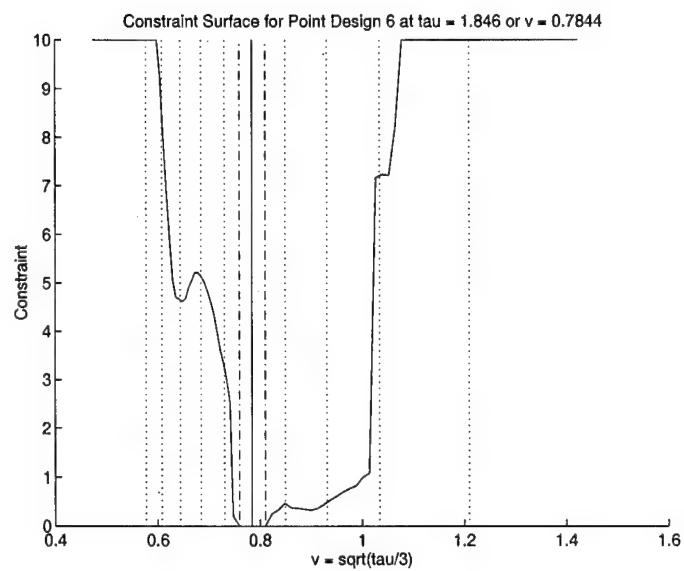
Figure D.16   Constraint Surface for Point Controller 6
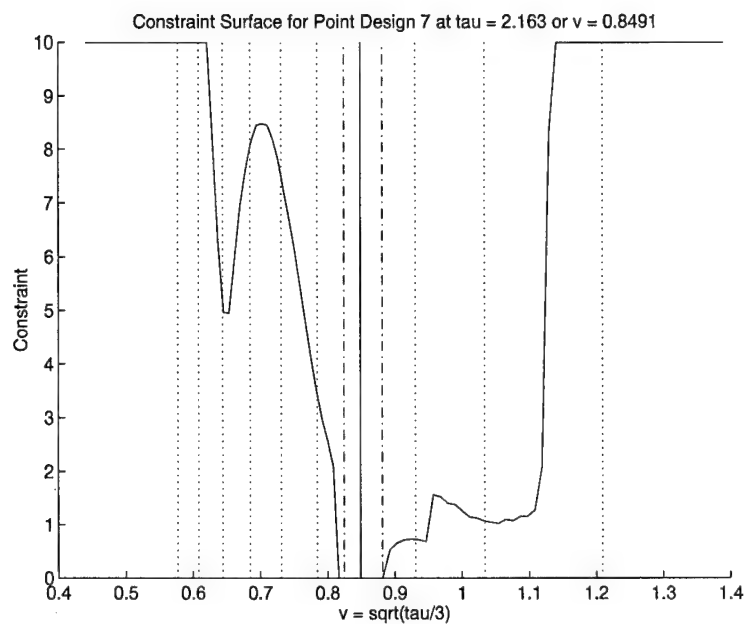
Figure D.17   Constraint Surface for Point Controller 7
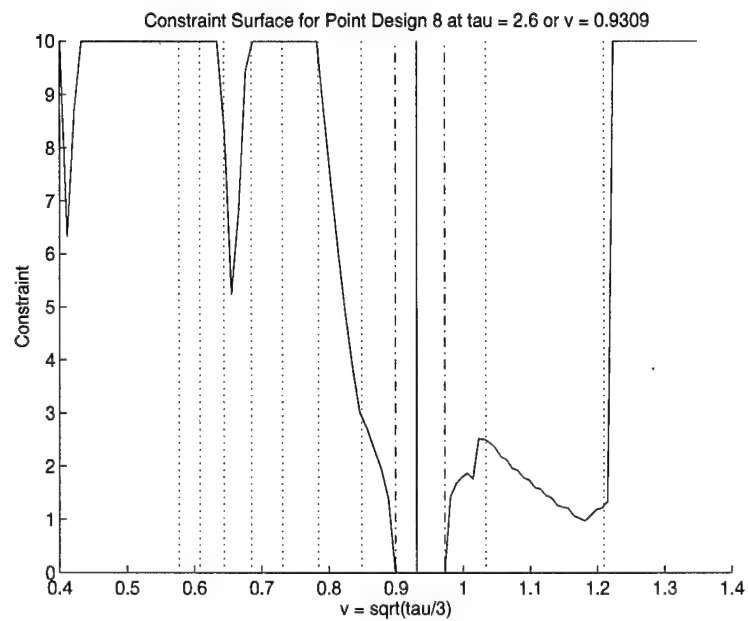
Figure D.18   Constraint Surface for Point Controller 8
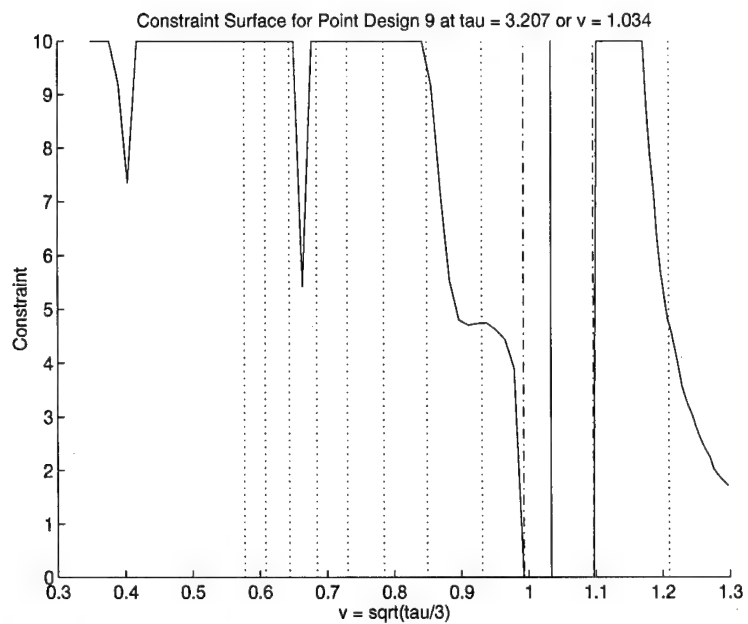
Figure D.19   Constraint Surface for Point Controller 9
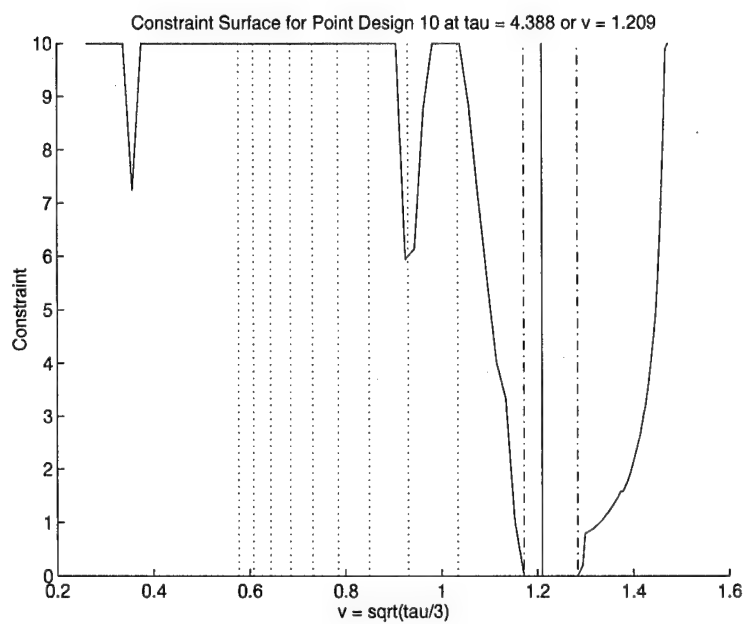
Figure D.20   Constraint Surface for Point Controller 10
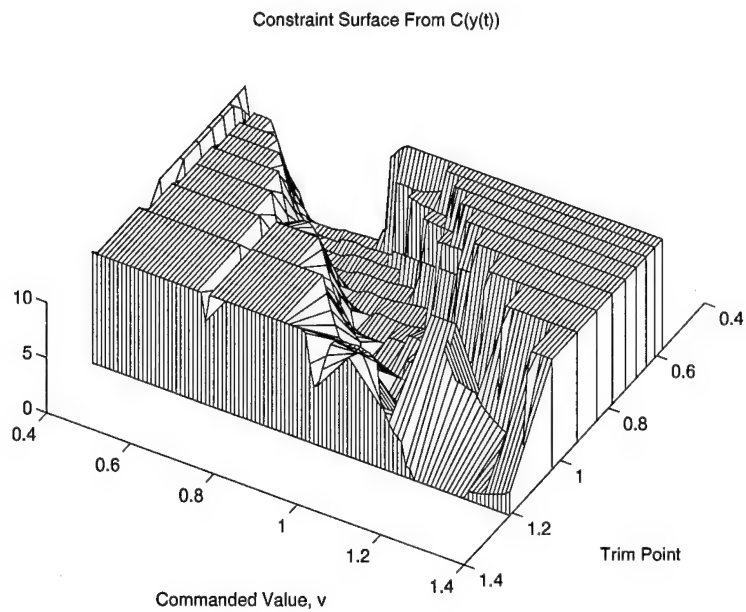
Figure D.21   Constraint Surface of Point Controllers, $C(y(t))$
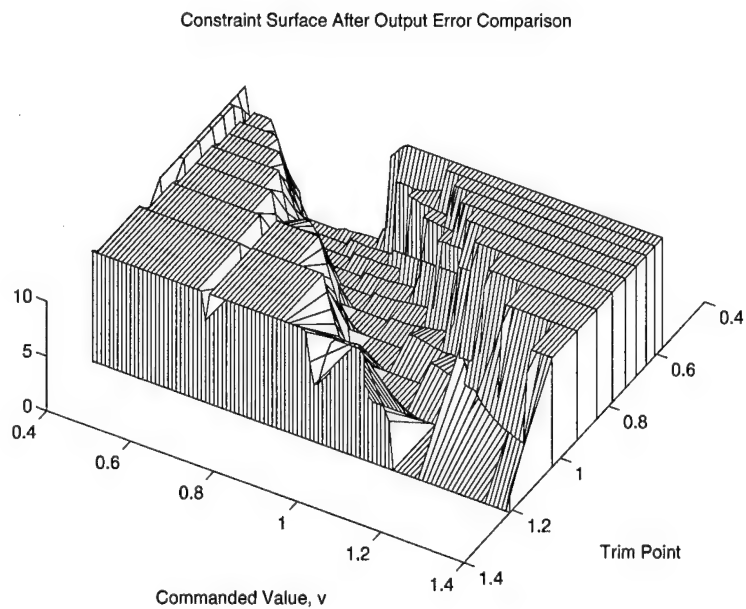
Figure D.22   Constraint Surface of Point Controllers after Normalized Output Error Check

D.3 Constraint Surface Plots of Scheduler: Nonlinear System
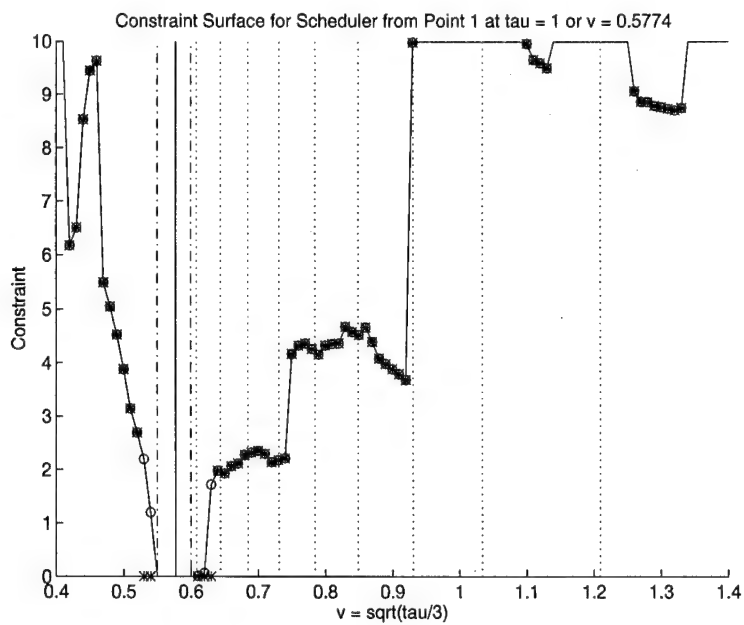
Figure D.23   Constraint Surface for Scheduler from Point 1
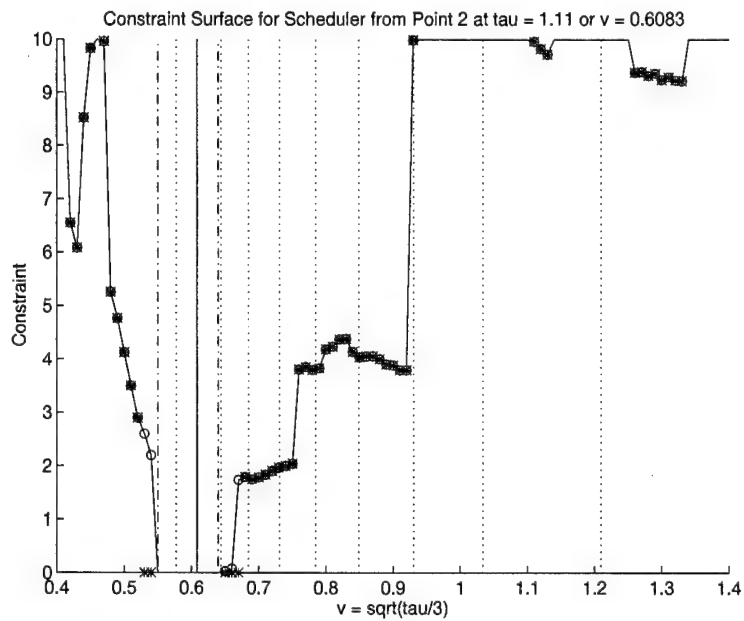
Figure D.24   Constraint Surface for Scheduler from Point 2
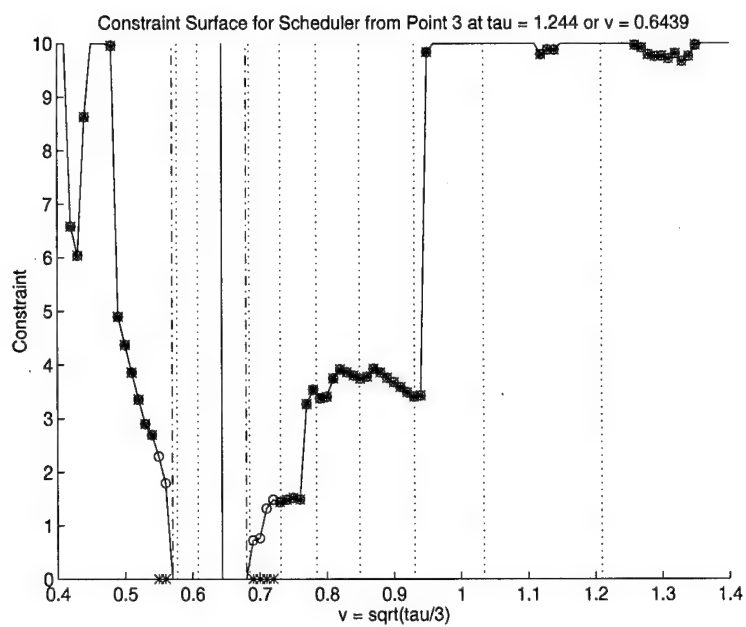
Figure D.25   Constraint Surface for Scheduler from Point 3
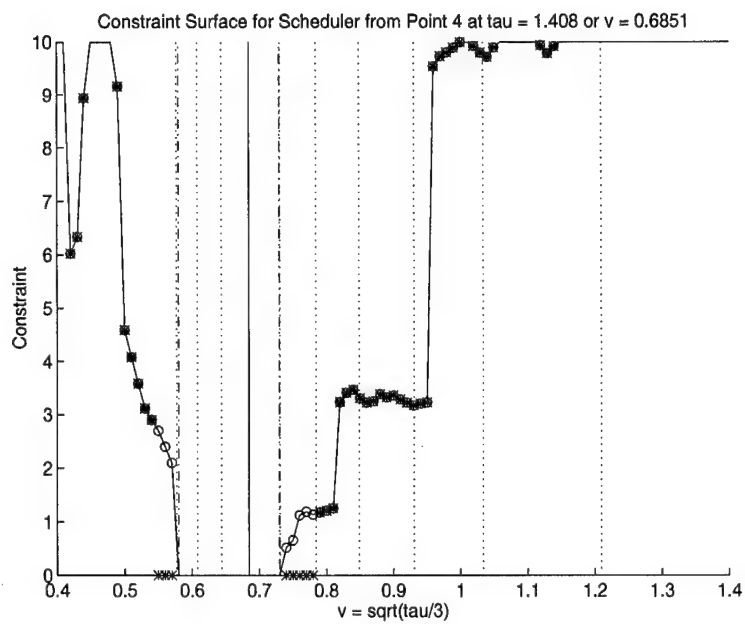
Figure D.26   Constraint Surface for Scheduler from Point 4
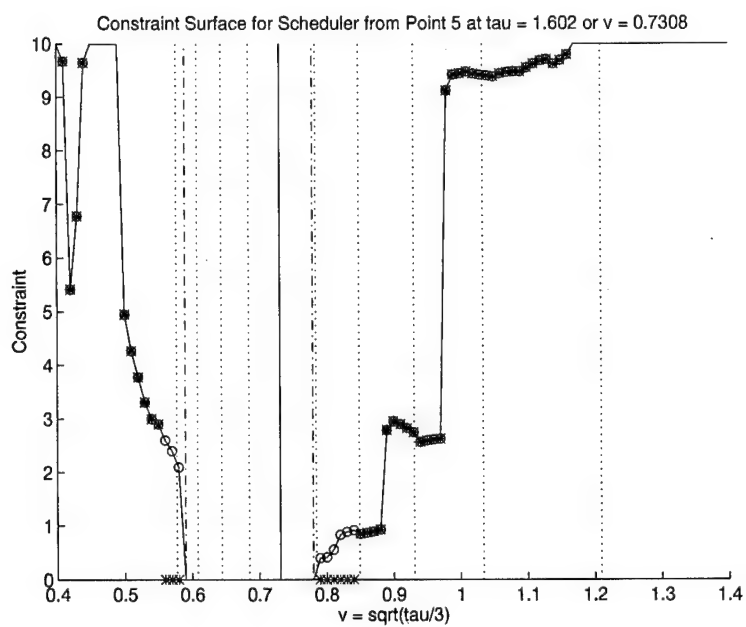
Figure D.27   Constraint Surface for Scheduler from Point 5
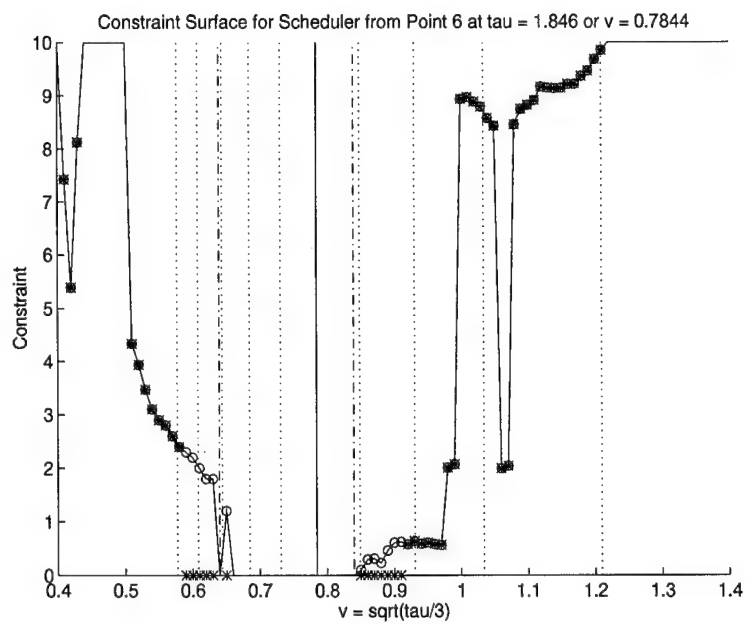
Figure D.28   Constraint Surface for Scheduler from Point 6

D-14
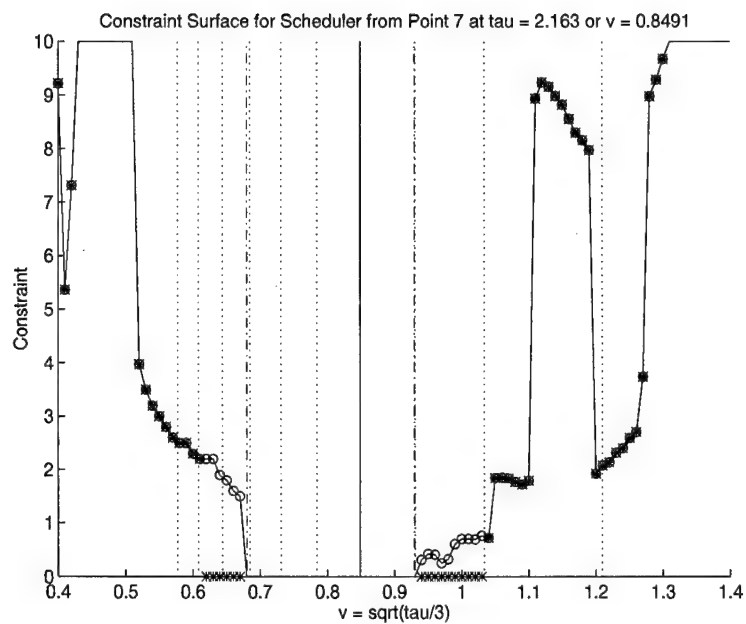
Figure D.29 Constraint Surface for Scheduler from Point 7
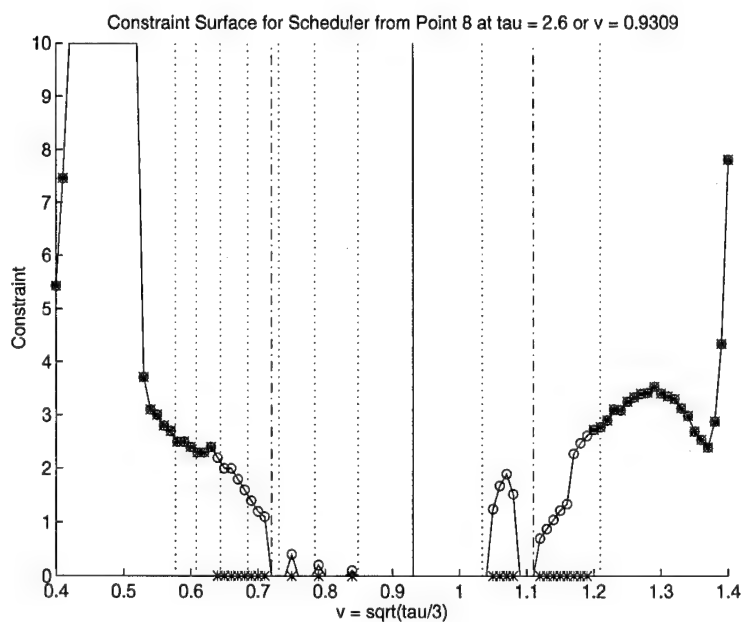
Figure D.30 Constraint Surface for Scheduler from Point 8
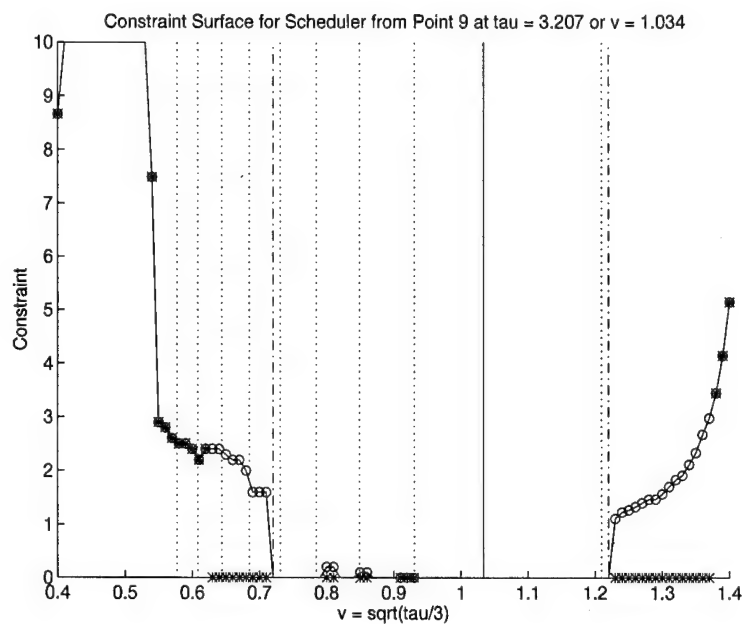
Figure D.31   Constraint Surface for Scheduler from Point 9
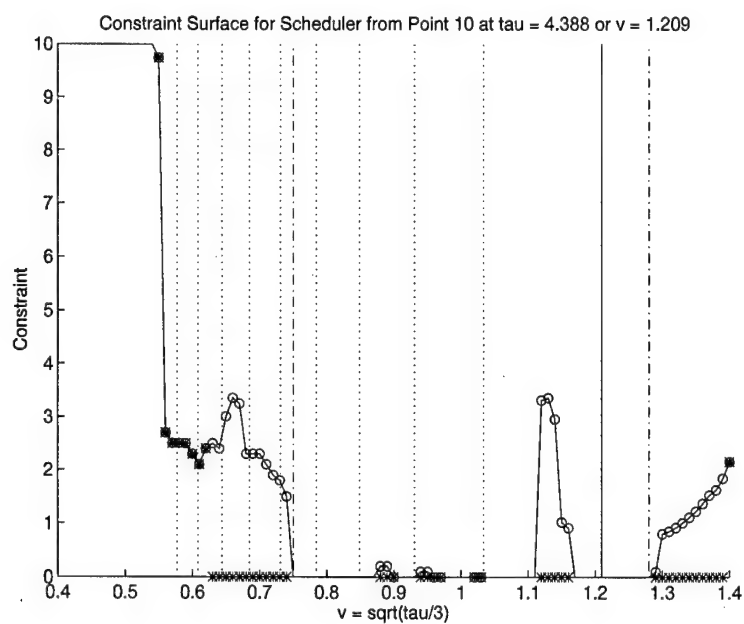
Figure D.32   Constraint Surface for Scheduler from Point 10

D-16

Constraint Surface From C(y(t))
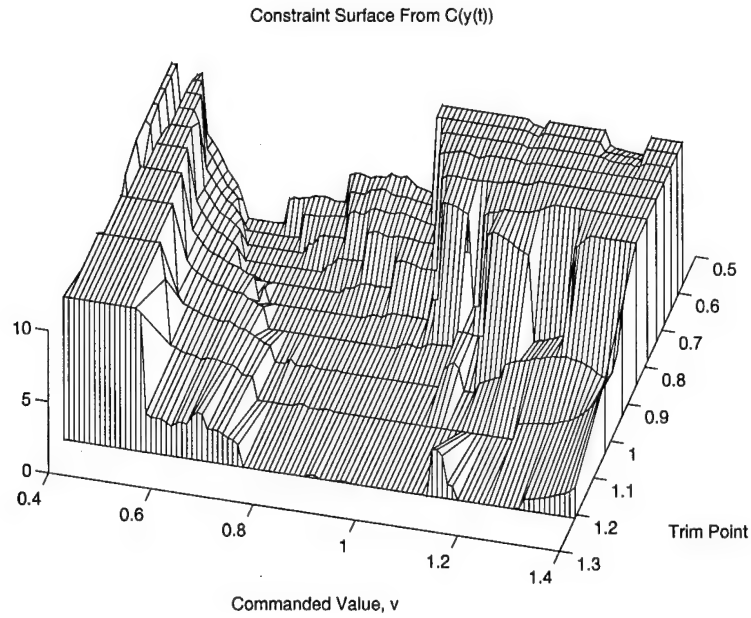
Figure D.33   Constraint Surface of Scheduler, $C(y(t))$

Constraint Surface After Output Error Comparison
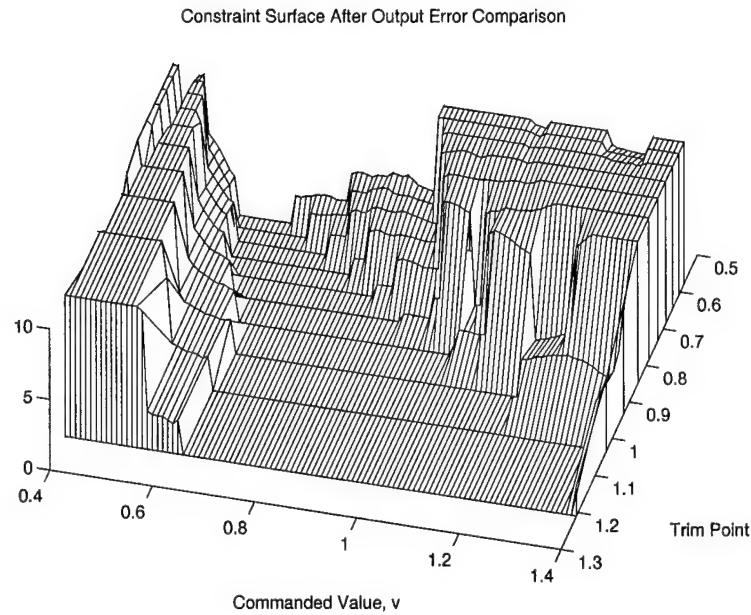
Figure D.34   Constraint Surface of Scheduler after Normalized Output Error Check

D-17

D.4 Increase in Cover by Scheduler: LTV System
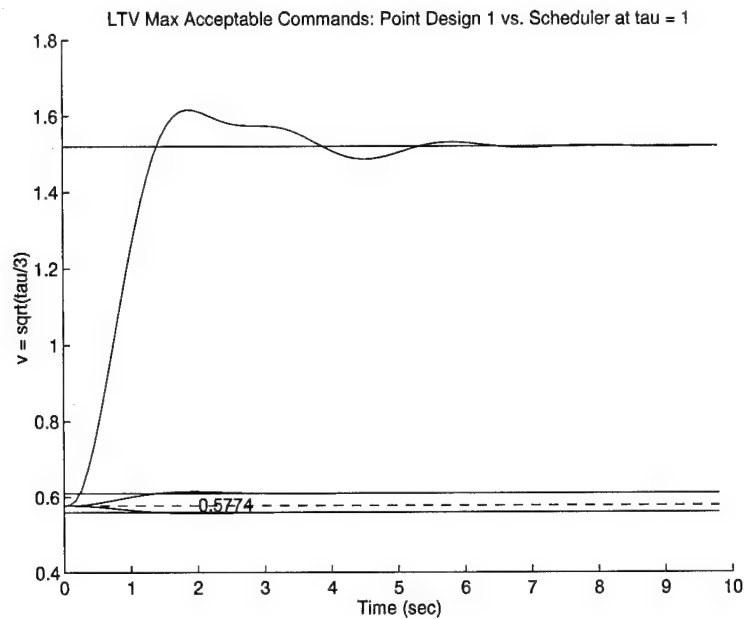
Figure D.35 Increase in Slewing Capability from Point Design 1
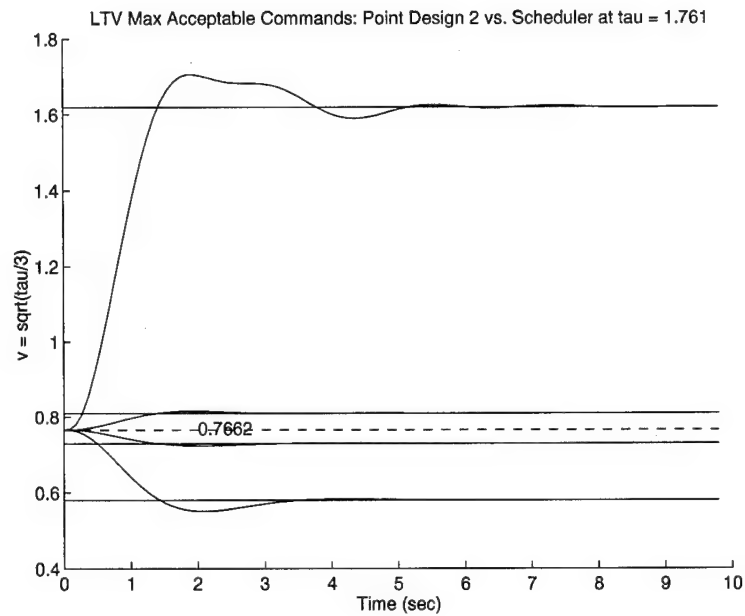
Figure D.36 Increase in Slewing Capability from Point Design 2
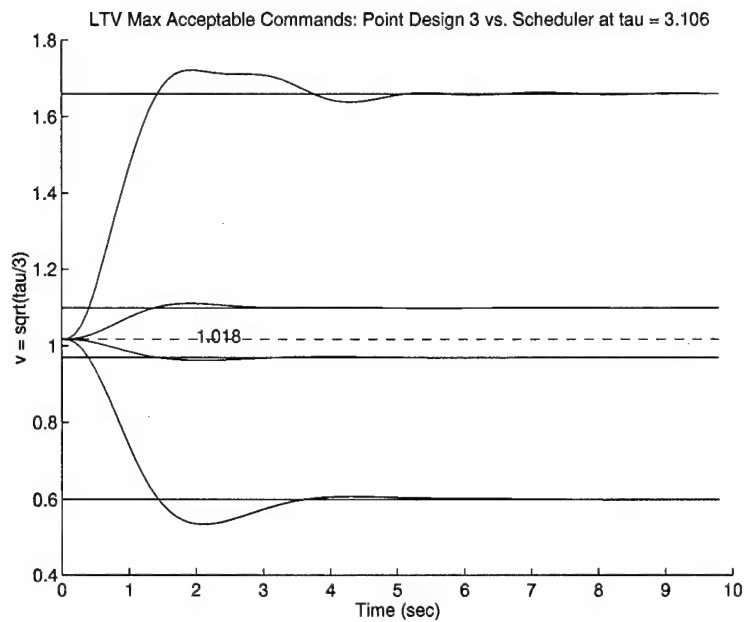
Figure D.37   Increase in Slewing Capability from Point Design 3
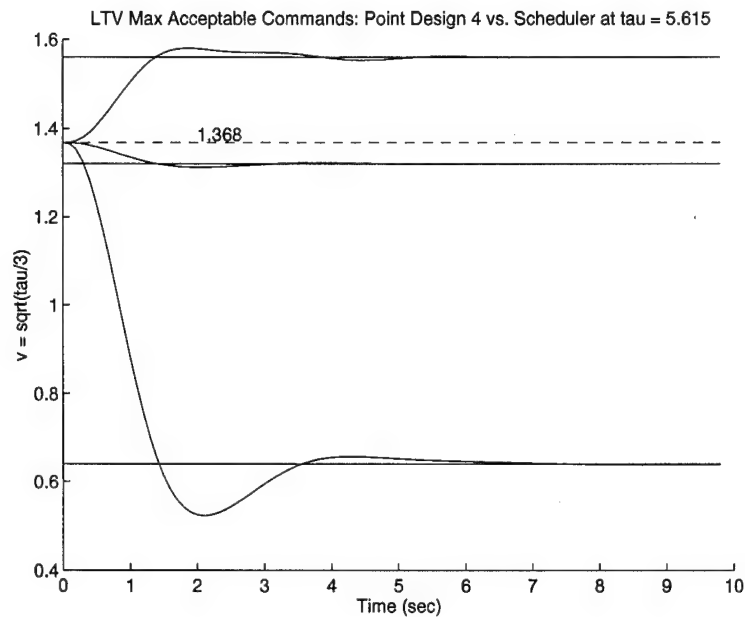
Figure D.38   Increase in Slewing Capability from Point Design 4

D-19

D.5   Constraint Surface Plots of Point Controllers: LTV System
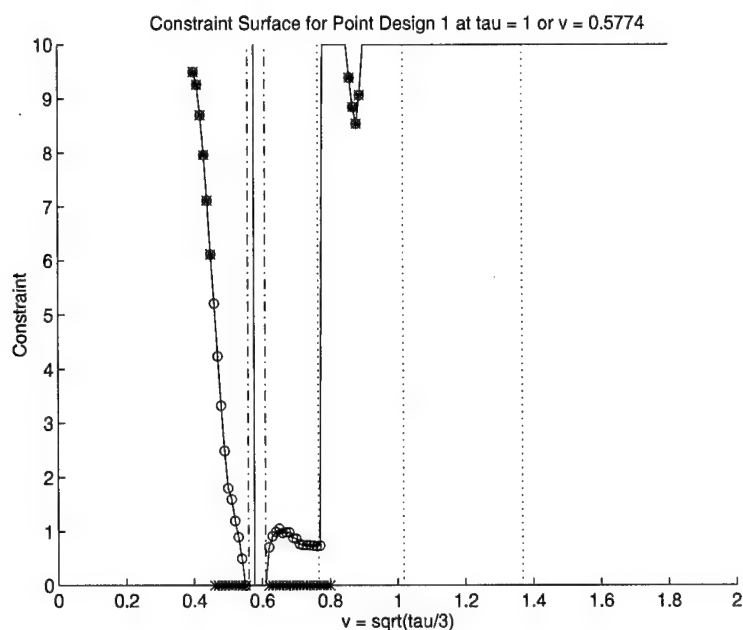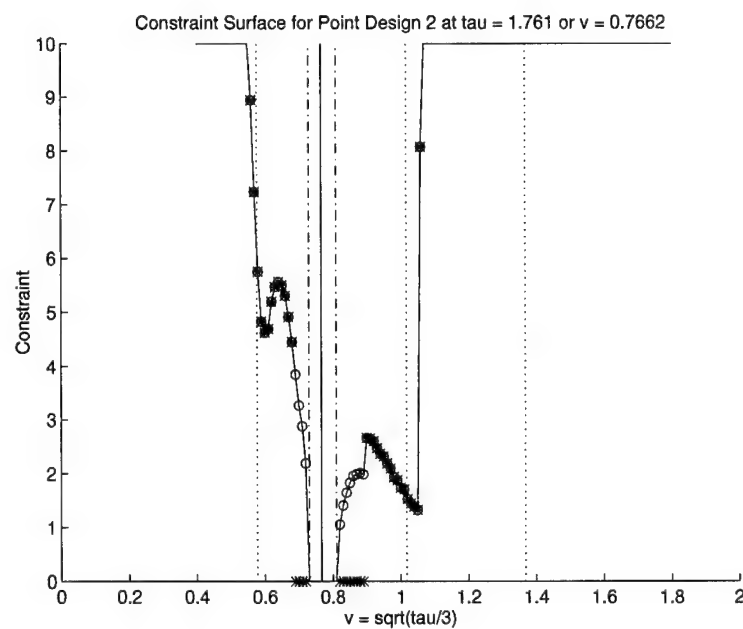

Figure D.39   Constraint Surface for Point Controller 1

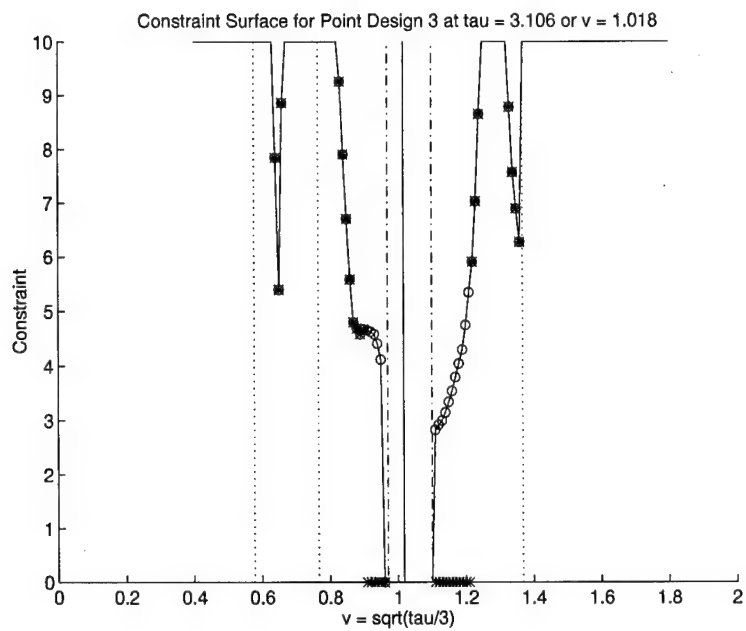Figure D.40   Constraint Surface for Point Controller 2

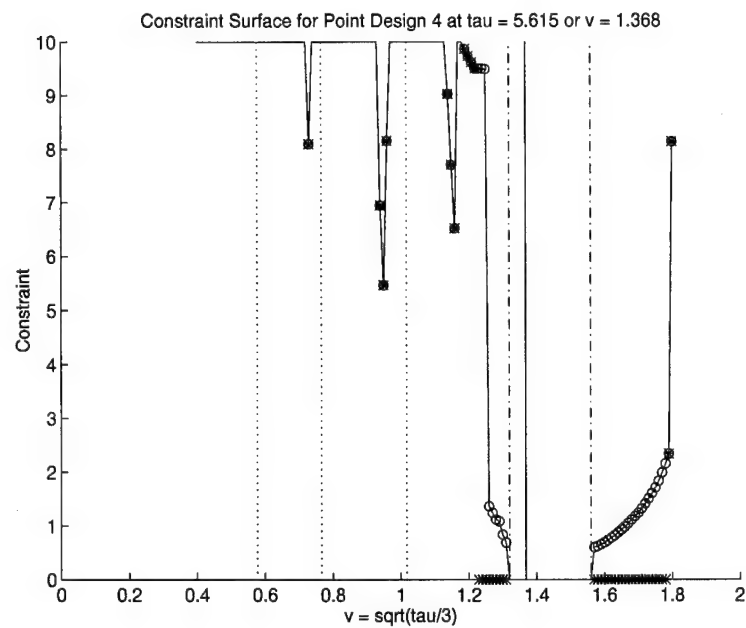Figure D.41   Constraint Surface for Point Controller 3

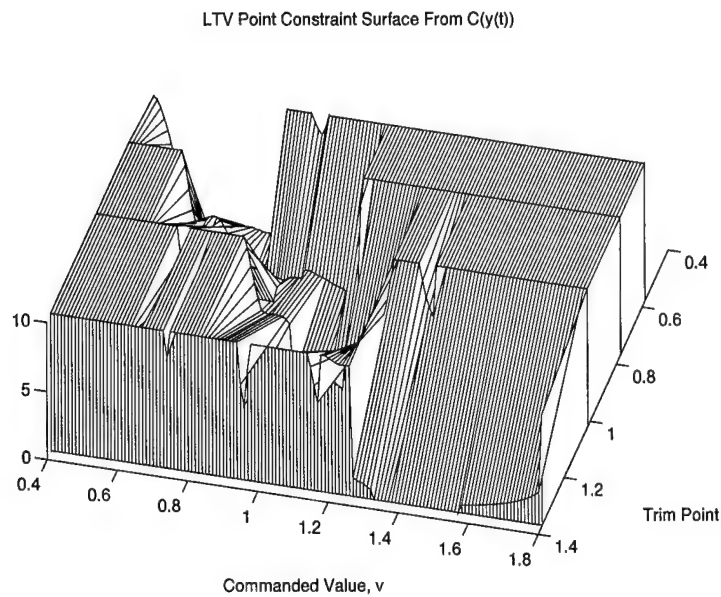Figure D.42   Constraint Surface for Point Controller 4

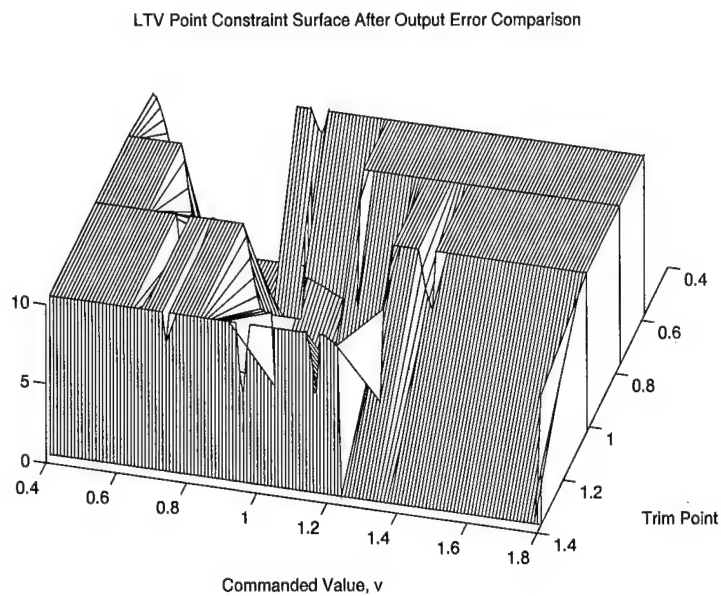Figure D.43   Constraint Surface of Point Controllers, $C(y(t))$

Figure D.44   Constraint Surface of Point Controllers after Normalized Output Error
Check

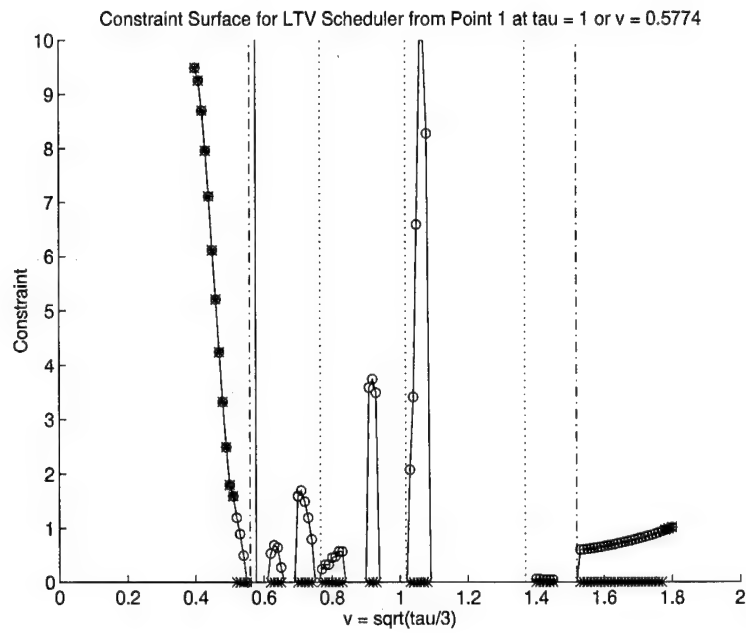## D.6 Constraint Surface Plots of Scheduler: LTV System

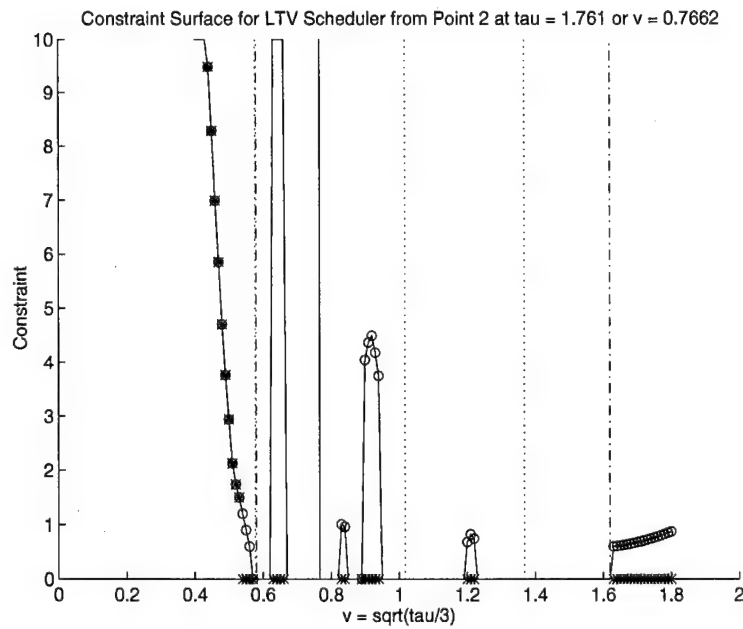Figure D.45 Constraint Surface for Scheduler from Point 1

Figure D.46 Constraint Surface for Scheduler from Point 2
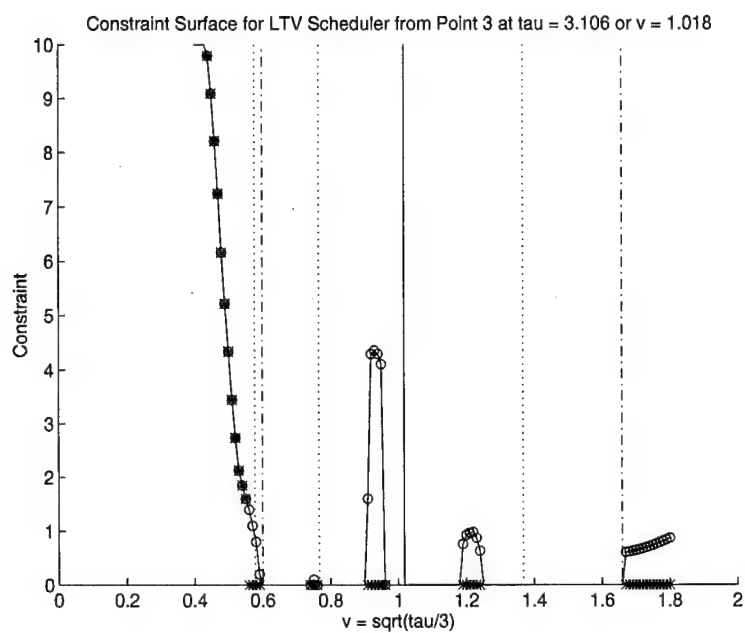
Figure D.47   Constraint Surface for Scheduler from Point 3

Figure D.48   Constraint Surface for Scheduler from Point 4

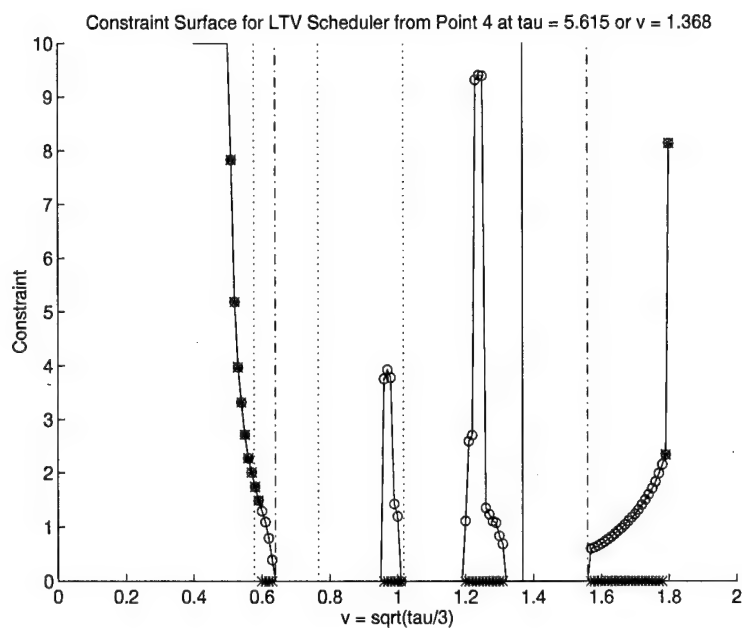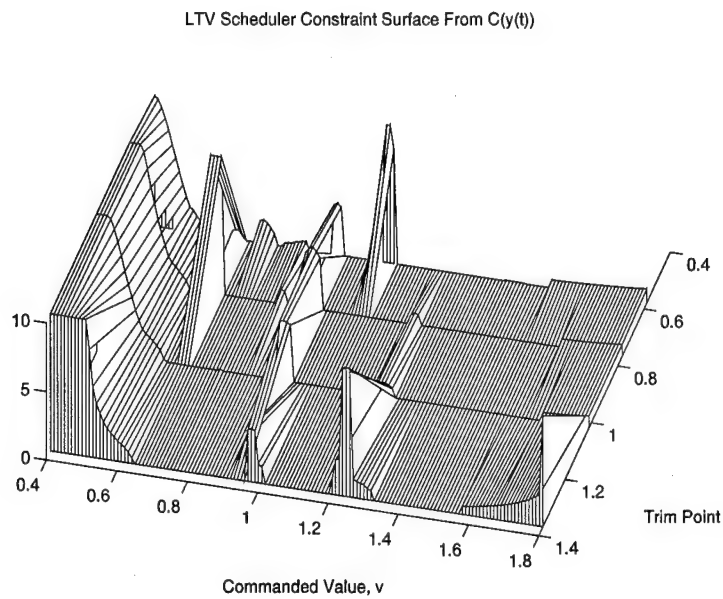LTV Scheduler Constraint Surface From C(y(t))

Figure D.49   Constraint Surface of Scheduler, $C(y(t))$

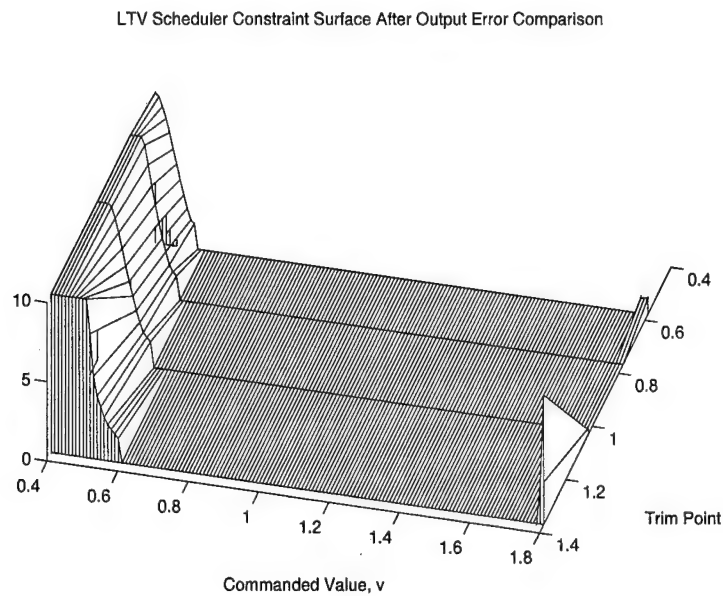LTV Scheduler Constraint Surface After Output Error Comparison

Figure D.50   Constraint Surface of Scheduler after Normalized Output Error Check

Bibliography

1. Azvine, B. and R.J. Wynne "Improved MIMO Quantitative Feedback Design Using a Matlab Toolbox," *1995 QFT Symposium*, (August 1995)

2. Batur, C., and A. Srinivasan "Inverse Fuzzy Model Controllers," *Proceedings of the American Control Conference*, 772-76 (June 1993).

3. Bennett, Stuart "Development of the PID Controller," *IEEE Control Systems Magazine, 13:6* 58-65 (December 1993).

4. Berenji, H. R. and P. Khedkar "Learning and Tuning Fuzzy Logic Controllers Through Reinforcements," *IEEE Transactions on Neural Networks, 3:5* 724-40 (September 1992).

5. Berenji, H. R. and P. Khedkar "Adaptive Fuzzy Control with Reinforcement Learning," *Proceedings of the American Control Conference*, 1840-45 (June 1993).

6. Betzold, Capt Robert W. *Multiple Input - Multiple Output Flight Control Design with Highly Uncertain Parameters; Application to the C-135 Aircraft.* MS-Thesis, AFIT/GE/EE/83D-11. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, December 1983.

7. Blakelock, J.H. *Automatic Control of Aircraft and Missiles.* (Second Edition), New York: John Wiley & Sons, 1991.

8. D'Azzo, J.J. and C.H. Houpis *Linear Control System Analysis & Design.* (Fourth Edition), New York: McGraw-Hill, 1995.

9. Fortune, Steven "Voronoi Diagrams and Delaunay Triangulations", *Computing in Euclidean Geometry.* 193-233, Singapore: World Scientific Publishing Co., 1992

10. Gamble, Capt Lynn L. and Capt William L. Smith *Improved Dutch Roll Stability Augmentation System for a Modified C-135B Aircraft.* MS-Thesis, AFIT/GGC/EE/70-8. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, May 1970.

11. Goldberg, David E. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading MA: Addison-Wesley Pub. Co., 1989

12. Griffin, J.M. *Analog Simulation of a Modified C-135 Aircraft Having a Series Yaw Damper.* Aeronautical Systems Division Technical Report ASD-TR-69-97. Wright-Patterson AFB OH: Air Force Systems Command, January 1970 (AD 867411L)

13. Griffin, J.M. *Digital Computer Solution of Aircraft Longitudinal and Lateral-Directional Dynamic Characteristics.* Aeronautical Systems Division, Deputy for Engineering Technical Report SEG-TR-66-52. Wright-Patterson AFB OH: Air Force Systems Command, December 1967 (ADA 078672)

14. Guibas, L. and J. Stolf "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams", *ACM Transactions on Graphics, Vol. 4 No. 2,* 74-123 (April 1985)

15. He, S., S. Tan, F. Xu and P. Wang. " PID Self-Tuning Control using a Fuzzy Adaptive Mechanism," *IEEE Conference on Fuzzy Systems, Vol2* 708-13 (1993).

16. Horowitz, Isaac M. *Quantitative Feedback Design Theory.* Boulder CO: QFT Publications, 1992

17. Hung, C-C and B. Fernandez. "Comparative Analysis of Control Design Techniques for a Cart-Inverted-Pendulum in Real-Time Implementation," *Proceedings of the American Control Conference,* 1870-73 (June 1993).

18. Jang, J. R. "Self-Learning Fuzzy Controllers Based on Temporal Back Propagation," *IEEE Transactions on Neural Networks, 3:5* 714-23 (September 1992).

19. Karr, C. L. and E. J. Gentry. "Fuzzy Control of pH Using Genetic Algorithms," *IEEE Transactions on Fuzzy Systems, 1:1* 46-53 (February 1993).

20. Kobylarz, T.J., M. Pachter and C.H. Houpis, "Fuzzy Identification in Control Systems," *World Scientific Series In Applicable Analysis vol 5: Recent Trends in Optimization Theory and Applications.* Singapore, To appear in July 1995

21. Kosko, Bart *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence.* Englewood NJ: Prentice Hall, 1992

22. Lai, J and Y. Lin. "Fuzzy Model-Based Control of a Pneumatic Chamber," *Proceedings of the American Control Conference,* 1162-66 (June 1993).

23. Layne, J. R., K. Passino and S. Yurkorich. "Fuzzy Learning Control for Antiskid Braking Systems," *IEEE Transactions on Control Systems Technology, 1:2* 122-29 (June 1993).

24. Lee, D.T. and F.P. Preparata "Computational Geometry – A Survey", *IEEE Transactions on Computers, Vol. c-33 No. 12,* 1072-11011 (December 1984)

25. Logan, Capt Michael W. *Investigation into Model-Based Fuzzy Logic Control.* MS-Thesis, AFIT/GE/ENG/93D-25. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, December 1993.

26. Logan, Michael W. and Meir Pachter "Full-Envelope Fuzzy Logic Control" *NAECON,* 598-605 (May 94)

27. Luenberger, D.G. *Optimization By Vector Space Methods*. New York: John Wiley & Sons, 1969.

28. *MATLAB Optimization TOOLBOX*. Natick MA: The MathWorks Inc., 1992

29. *MATLAB Reference Guide*. Natick MA: The MathWorks Inc., 1992

30. Maybeck, Peter S. *Stochastic Models, Estimation and Control, Vol 3*, New York NY: Academic Press, 1982.

31. Martinez, T. and K. Schulten "Topology Representing Networks", *Neural Networks, Vol. 7 No. 3*, 507-522 (1994)

32. Nie, J. and D.R. Linkens "Automatic Knowledge Acquisition for Multivariable Fuzzy Control Using Neural Network Approach," *Proceedings of the American Control Conference,* 767-71 (June 1993).

33. Pachter, Meir and D.H. Jacobson, "The Stability of Planar Dynamical Systems Linear in Cones," *IEEE Transactions on Automatic Control, 26:2* 587-590 (April 1981).

34. Pachter, Meir, S. Sheldon, and M. Mears. *Intelligent Flight Control* Technical Report, USAF Flight Dynamics Laboratory, 1995 (distribution pending)

35. Rueda, A. and W. Pedrycz. "Fuzzy Coordinator in Control Problems," *North American Fuzzy Information Processing Society '92* 322-29, NASA, Houston TX (December 1992).

36. Rueda, A. and W. Pedrycz. "A Design Method for a Class of Fuzzy Hierarchical Controllers," *IEEE Conference on Fuzzy Systems, Vol1* 196-99 (1993).

37. Shamos, I. and D. Hoey "Closest-Point Problems", 16th *Annual Symposium on Foundations of Computer Science,* 151-162 (October 1975)

38. Shoureshi, R., P. Torcellini, and K. Rahmani. "Derivation and Implementation of Fuzzy Optimal Control," *Proceedings of the American Control Conference,* 1860-64 (June 1993).

39. Special Issue of Neural Network Hardware, *IEEE Transactions on Neural Networks, 4:3* (May 1993).

40. Smith, S. M., and D. J. Comer. "Automated Calibration of a Fuzzy Logic Controller Using a Cell State Space Algorithm," *IEEE Control Systems Magazine,* 18-28 (August 1991).

41. Toussaint, G.T. "Pattern Recognition and Geometric Complexity", *IEEE Transaction on Pattern Recognition, Vol. 2*, 1324-1347 (1980)

42. Vachtsevanos, G., S. S. Farinwata, and D. K. Pirovolou. "Fuzzy Logic Control of an Automotive Engine," *IEEE Control Systems Magazine,* 62-68 (June 1993).

43. Wang, L. "Stable Adaptive Fuzzy Control of Nonlinear Systems," *IEEE Transactions on Fuzzy Systems, 1:2* 146-55 (May 1993).

44. Wang, L. and M. Mendel. "Generating Fuzzy Rules by Learning from Examples," *Proceedings of the IEEE Conference on Intelligent Control,* 263-68 (August 1991).

45. Wang, L. and M. Mendel. "Fuzzy Adaptive Filters, with Application to Nonlinear Channel Equalization," *IEEE Transactions on Fuzzy Systems, 1:5* 161-70 (August 1993).

46. Yamakawa, Takeshi. "A Fuzzy Inference Engine in Nonlinear Analog Mode and its Application to a Fuzzy Logic Controller," *IEEE Transactions on Neural Networks, 4:3* 496- 522 (May 1993).

47. Yen, J., F. Wang and Y. Chen. "A Fuzzy Scheduling Controller for the Computer Disk File Track- Following Servo," *IEEE Conference on Fuzzy Systems, Vol2* 1016-21 (1993).

48. Zimmermann, Hans J. *Fuzzy Set Theory and its Applications.* Boston MA: Kluwer Academic Press, 1991

49. Zhao, Tienan and T. Virvalo. "Fuzzy Control of a Hydraulic Position Servo with Unknown Load," *IEEE Conference on Fuzzy Systems, Vol2* 785-88 (1993).

50. Zhao, Zhen-Yu, M. Tomizuka and S. Isaka "Fuzzy Gain Scheduling of PID Controllers," *IEEE Transactions on Systems, Man, and Cybernetics 23:5* 1392-98 (October 1993).

Vita

Captain Thomas J. Kobylarz was born on 21 July 1963 in Trenton, New Jersey. He graduated from high school in Ewing Twp., New Jersey in 1981 and attended the United States Air Force Academy from which he received the degree of Bachelor of Science in Electrical Engineering in May 1986. Upon graduation he received a regular commission in the Air Force and attended undergraduate pilot training at Laughlin AFB Texas from July 1986 to June 1987. In June of 1987 he entered the School of Engineering, Air Force Institute of Technology, WPAFB OH. In December 1988 he received the degree of Masters of Science in Electrical Engineering and was the recipient of the school's Mervin E. Gross award. From December 1988 to June 1992 he was assigned to the 4950th Test Wing, WPAFB Ohio working on the Airborne Radar Testbed. In June of 1992 he again entered AFIT's School of Engineering, this time working towards his Ph.D. in the Department of Electrical and Computer Engineering, entering candidacy March of 1994.

Captain Kobylarz is a member of the Tau Beta Pi and Eta Kappa Nu honor societies and has been a member of the IEEE since 1984. Following his graduation from AFIT he will be assigned to the AWACS Program Office, Hanscom AFB, Massachusetts.

Permanent address: 263 Pine Hill Road
Hollis, New Hampshire 03049

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1995	Doctoral Dissertation

4. TITLE AND SUBTITLE

FULL ENVELOPE CONTROL OF NONLINEAR PLANTS WITH PARAMETER UNCERTAINTY BY FUZZY CONTROLLER SCHEDULING

5. FUNDING NUMBERS

6. AUTHOR(S)

Thomas J. Kobylarz, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
2750 P Street
Wright-Patterson AFB, OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/DSG/ENG/95S-05

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

WL/FIGS-2
Wright-Patterson AFB, OH 45433-7521

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A full envelope controller synthesis technique is developed for multiple-input single-output (MISO) nonlinear systems with structured parameter uncertainty. The technique maximizes the controller's valid region of operation, while guaranteeing pre-specified transient performance. The resulting controller does not require on-line adaptation, estimation, prediction or model identification. Fuzzy Logic (FL) is used to smoothly schedule independently designed point controllers over the operational envelope and parameter space of the system's model. These point controllers are synthesized using techniques chosen by the designer, thus allowing an unprecedented amount of design freedom. By using established control theory for the point controllers, the resulting nonlinear dynamic controller is able to handle the dynamics of complex systems which can not otherwise be addressed by Fuzzy Logic Control. An analytical solution for parameters describing the membership functions allows the optimization to yield the location of point designs: both quantifying the controller's coverage, and eliminating the need of extensive hand tuning of these parameters. The net result is a decrease in the number of point designs required. Geometric primitives used in the solution all have multi-dimensional interpretations (convex hull, ellipsoid, Voronoi/Delaunay diagrams) which allow for scheduling on n-dimensions, including uncertainty due to nonlinearities and parameter variation. Since many multiple-input multiple-output (MIMO) controller design techniques are accomplished by solving several MISO problems, this work bridges the gap to full envelope control of MIMO nonlinear systems with parameter variation.

14. SUBJECT TERMS

Fuzzy Logic, Nonlinear Control, Structured Uncertainty, Scheduling, Gain Scheduling, Control, Voronoi, Delaunay

15. NUMBER OF PAGES

209

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL